March 2009

The Affects of Open Source Software Licenses on Business Software

Devin W. Kelly
Worcester Polytechnic Institute

Gregory K. Holtorf
Worcester Polytechnic Institute

Project Number: 48-KJR-COP5

The Affects of Open Source Software Licenses on Business Software

An Interactive Qualifying Report

Submitted to the Faculty

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for

the Degree of Bachelor Science

by

Devin Kelly _____

Greg Holtorf _____

Date: March 6, 2009

_____

Professor Kent J. Rissmiller, Advisor

1 Abstract

This project investigates open and closed source licenses and how each style of license changes the way business is done. We look at: license violations and consider techniques for reducing violations, why businesses use closed source and open source licenses, and the intellectual property rights businesses and individuals have in open source licenses. We show that open licenses are easier to enforce, but require a different business model and show how piracy can be effectively reduced.

Contents

2 Introduction

Free and open source software is a growing field in the software industry. In this project we look at the difference between commercial software, open source software, and other licensing models. We look at how enforceable the different models are, how easily they are distributed, how they are affected by digital rights management, and how profitable the different models are. Specifically, but we look to see how enforceable the various types of licenses are, how licenses affect the distribution and popularity of a product, how licenses affect code ownership, and how various licenses are affected by piracy and digital rights management. Additionally, we will look at how different licenses can enhance or hinder a firm's ability to receive compensation for its work. We analyze each of these models in depth and suggest methods for determining the appropriate license a given business model. We try to put our suggestions in terms of the commercial realities of today, like piracy, enforceability of licenses, digital rights management and the ability to make a profit.

In order to understand the project fully we must better define the licenses we are examining. For our purposes, commercial software is the traditional method of selling software. Commercial software is sold under a license which gives the end user the right to use a copy of the software, but not to own a copy. Sometimes the copy can be used permanently and sometimes it can be for a limited time. Traditionally the source code is not viewable and is considered both copyrighted and a trade secret. Occasionally developers can gain access to the source code after paying a fee and signing non-disclosure agreements.

Open source software is a new way of licensing software. Software is considered open source only if the license allows: 1) the users to freely redistribute the software. 2) Derived works are allowed. 3) The source code is included with the compiled code. 4) Several other trivial criteria are met.[1] This is important because an open source license preserves the software from going into the public domain and ensures the developer maintains some rights to the software.

There are many other licenses that do not fall completely under the definition of commercial software or open source software. Some of the licenses are more or less restrictive than the open source model. Some licenses strike a balance between commercial software and open source, while some licenses give the user almost complete control of the product. We attempt to cover both

traditional commercial software as well as open source software. Under the circumstances, we cannot fully cover the effects of all other kinds of licenses but they will be referred to often.

Open source licenses have grown not only in popularity, but in variation of licenses. The first significant open source license was Richard Stallman's General Public License, which was developed in late 1980s. Since Stallman's original license was released it had gone through several versions, had several spin off licenses created, such as Lesser General Public License (LGPL), and many other similar open source licenses have been created. These additional open source licenses include the BSD license and the Creative Commons license. Open source licensing is approaching a point where it is becoming impossible to ignore for both closed source software developers and users of software. Closed source stalwart Microsoft has even started an open source division, and created its own open source license, the Microsoft Public License (Ms-PL)[2]. International Business Machines (IBM) has had a larger role in the development of open source projects, such as the Eclipse integrate development environment (IDE) and the Jazz collaboration environment. The open source movement starts with one man and one license and has grown to the scale of multinational corporations, some of which are notorious for their close source practices, and hundreds of different licenses.

Recommendations will be provided in the sections below regarding software licensing. The recommendations will provide insight on selecting the appropriate license for the software developed. These recommendations will consider software applications, how the license can be enforced, marketing, techniques for acquiring compensation, and the goal of the creator of the software.

This project aims to address the legal implications of the different software licenses that exist. The aim is to show how they affect software as a business and how a software creator, whether an individual or business, can use proper licensing to protect their rights to the software as creators and maximize the softwares effectiveness in the marketplace.

3 Background

Enforceability is directly related to the terms of the license. Licenses tend to restrict the reselling, redistributing, derived works, and personal use of a software product. For instance, commercial

software often disallows derived works, redistribution, and reselling of a product, while enforcing strict limits on what a user can do with the software. Licenses must be accepted before the software may be used. Most open source licenses allow redistribution and derived works (under specific circumstances). Some also disallow the reselling of the product or derived works. Open source software is given as a product rather than a license, which means that a user 'owns' the copy of the software on his or her computer, it is not licensed to them. [1] Licenses in open source software only need to be accepted when the software is altered.

The more free licenses are easier to enforce and the more restrictive licenses are difficult to enforce. As mentioned above, the enforcement of licenses can be divided into roughly two categories: enforcing open licenses and closed licenses. Before licenses can be enforced, the infringers on those licenses must be identified. Those who infringe on open licenses are typically companies who violate some term of the license. For example, editing software under an open license, such as GNU's Not Unix General Public License (GNU GPL, or GPL), and not making those changes available violates the license. These incidents happen infrequently, but when they do occur they are significant. An incident of this nature did occur in 2007 when the MPAA released a University Toolkit.[2] The toolkit was based off the Ubuntu Linux operating system, which is licensed under GPL. The changes were not made available, and the software was issued a Digital Millennium Copyright Act (DMCA) take down notice from Canonical, the makers of Ubuntu. The DMCA is a bill that was passed by the US Congress in 1998 in order to adapt copyright law to new technologies. The In this example, a company was the infringer on a license and it was simple for the maker of the original software to have the infringing software removed. The other situation, when closed software is infringed upon, is harder for the owner to enforce. The license enforcement for closed software is complementary to open software. When enforcing open software, companies are typically the infringers, but enforcing closed software the infringers are typically individuals, or end users. It is much more difficult for the owners of closed software to enforce their licenses as they must find and prosecute infringing individuals.

There are many different kinds of open source licenses, because many companies develop their own license to distribute under. These licenses may be based on another license, we will not attempt

to identify all licenses, just the most popular and well known open source licenses.

The GNU General Public License or GPL is one of the most popular OSS licenses. It follows the open source definition and adds several other sections, for instance it requires that attribution be given to the original software when a derivative work is created and it must be explicitly stated that you modified the software. It requires that you pass on the same rights that the original work was created under. So for instance if the original work was noncommercial the derivative work must also be noncommercial. No warrantee is provided in a GPL license. Looking at the GPL license you see that it is very concerned with protecting the rights of the user and the rights of the license holder. [3]

Like the GPL, BSD licenses provide no warranty and require that that warning be attached to the product. BSD licenses follow the open source definition but have important differences from the GPL. BSD licenses are a template that allows the source code of the software to be modified or copied. A BSD license does not required that the derivative software be open source or free. The BSD license basically allows for the software being developed to be redistributed or changed freely while protecting the software developer from liability.[4]

The Creative Commons license is popular because it can be applied to music, art, or other IP as well as software. The Creative Commons license is considered to have some rights reserved and four conditions may or may not be applied to the license, based on the discretion of the creator. These are 1. Share Alike - The creator may allow others to redistribute the work under an identical license. 2. Attribution- The creator may require that if others copy, use, or modify the work, the creator is given credit in the way requested. 3. Noncommercial - The creator decides whether the work may be profited from. 4. Derivative Works- The creator decides if the work may be modified. [5]

There are many differences between closed and open licenses; they are nearly opposite one another. When one reads a closed source license the entire text will, for the most part, have a single theme: the text in a closed license restricts and limits the users' privileges regarding the software. When compared to an open license the text is opposite. Most open licenses explicitly give the user rights and privileges. As explained above, open licenses give the users rights, and

typically places only one restriction, the creator of the original software must be credited if another user modifies the original software.

There are still more differences between closed licenses and open licenses. One that will affect the analysis of closed licenses is the large array of closed licenses in comparison to open licenses. When considering open licenses there are a handful or archetypal licenses, BSD, GPL, and Creative Commons. There are also derivative licenses, such as Lesser GPL (LGPL), FreeBSD, or OpenBSD, where there are subtle differences between each license. However, closed licenses nearly all come from corporations with their own legal departments. This allows a company to tailor a license specifically to each software release and version the company develops. As one might expect a firm like Microsoft has hundreds of different licenses, one for each version of each software release. Similarly, any firm that releases software under a closed license writes the license for that specific piece of software. This makes analyzing closed licenses more complex since no one license is more popular than any other closed source license.

Since there are so many different closed source licenses analyzing a select few different licenses is not possible. This paper will instead analyze the common themes of closed source licenses and avoid the minutiae that lie in different closed source licenses. The themes that will be analyzed include restrictions on ownership, access to source code, replication of compiled code, and users' rights related to sharing the software in question. The approach taken by closed source licenses is nearly uniform for all of these topics. Closed source licenses typically forbid a user from owning the software purchased; they instead grant the right to use the software and specifically forbid the right to own the software. By definition closed source licenses forbid the user access to the source code therefore indirectly forbidding the user to modify the software in question. Closed source licenses additionally forbid a user to copy the software in any way. However, this is usually achieved via other methods. As described above most software considered in this paper was developed and compiled for x86 machines making it simple to copy the compiled coded and run that code on another x86 machine. Since the compiled code can be copied easily, closed source software vendors sometimes rely on DRM technologies as well as other techniques to prevent copying the compiled code.

Closed source licenses typically have more limitations in addition to disallowing copying compiled code and viewing source code. Licenses agreements can also limit the way software is used. For example, a closed source license agreement may forbid a user from executing the compiled code on certain machines or machines with certain hardware. Another typical limitation in closed source licenses is the revocation of rights not mentioned in the license. This allows the software vendor to reserve the rights on any topics not covered in the license.

The practice of using licenses to control reselling, redistributing, and of derived works is well established legally. Also recent litigation has show open source licenses to be enforceable. [6][7] However, personal use is a gray area in licensing. Restricting users' personal use of a software product through a license has been judged legal or illegal in multiple different court cases. [8] Because judges are often ignorant of copyright law, licensing law is inconsistent in lower courts. Aside from the standard self protection clauses, open source software sidesteps this issue by giving the user the freedom to use the software as they like.

The more restrictive licenses are more popular than the free licenses in many cases. We expect this to be due to that the fact that software protected by restrictive licenses are more developed. For example, device driver support for Windows is much better than Linux/Mac due to Windows' market share. When a third party sells a product, it go after the largest market and neglect smaller markets. This is especially true for Linux, as Apple tends to limit the hardware one can use with a Mac. This phenomenon is cyclical as well, when device developers only develop for Windows, Windows popularity only increases. Also, Windows is easier to support as Microsoft has the resources to develop or encourage others to develop drivers for popular hardware when hardware vendors do not release Windows drivers. A case study that could be used here would be ATI's refusal to support open platforms with device drivers.[9] Another case study illustrates the point. In the engineering industry, two related software frameworks called Matlab and Simulink are both incredibly popular in industry. Matlab is a high level (easy to learn) programming language, but the software framework comes with a series of toolboxes for specialized applications. However, there are open source, and free, alternatives to Matlab one called Octave and another called Scilab. The Octave language is over 90% compatible with the Matlab language however its market share

is a tiny fraction of Matlabs'.

The other side of enforcement is realistic enforcement or how well can a developer actually enforce their license? When it comes to how licenses affect reselling there is somewhat of a trade off. Thanks to the Internet, it is practically impossible to prevent your software from being stolen. The three methods to decrease the amount of theft are digital rights management, legal measures, and social measures. Legal measures have proven ineffective due to the cost and inability to actually catch any large amount of pirates. DRM and social measures have varying degrees of success. However, having a closed source model tends to prevent other companies from stealing your source code. On the other side open source software releases its source code, so if the source was taken and used in a commercial, closed source product it is often impossible to tell. However it is much easier for a developer to bring legal action against a corporation than a large group of people. These basic points also apply to redistribution and derived works if you consider them in terms of lost market share/profits. Personal use cannot actually be restricted without draconian software measures which, not coincidentally, tend to be unpopular with users.

Licenses have a large effect on how software is used. Because almost all software is licensed, the license is considered legally binding even if the user does not agree to it. This means that the license holder has the freedom to demand basically whatever he or she wants from the user. This brings us to the issue of licenses affecting sales. A license has a huge effect on a product's popularity. An overly restrictive license will often dissuade your customers from buying the product. [10] Also, the license determines whether the product can be given away. An inferior product can often gain significant market share because it is free. Keep in mind there are many different factors that affect the popularity of any software product including: marketing, age, and quality some of which may be more important in determining a products popularity, therefore there are no actual statistics on the effects of licensing.[11]

Code 'ownership' is complicated at the best of times and it becomes more complicated with open source software. Code ownership is governed by the license it is created under. Usually in open source the original creator will be attributed as the owner of the base code and anyone who modifies the code will be attributed as 'owning' the changes he makes. One can see how attribution becomes

complicated as a project grows. [12] However, not all licenses require attribution to be given (keep in mind these are rare). If you were so inclined, one could create a license that forced contributors to turn in their code anonymously, however I don't think anyone has actually created such a license (yet). The question can be examined with a case study of the Linux Kernel. Originally written by Linus Torvalds in 1991 the Linux Kernel has had thousands of contributors, both individuals, small companies, large corporations, and individuals with sponsors. The Linux Kernel is licensed under GPL. With all these contributors who own the Linux Kernel, ownership of works in question is not mentioned in the GPL. When there is no owner, liability from any damages the software creates may not be easy reconcile.

Understanding this project also requires a large amount of technical knowledge of software. At the core of this project lies the issue of source code and compiled code. Source code is the human readable code that a programmer writes. Source code is fed into a compiler and is changed or compiled into compiled code, or machine code that the computer can now run. This is code that the computer can read but humans cannot. Once the code is compiled it is difficult to impossible to translate back to source code, which means that finished programs can be sold to the user without the user being able to tell how they are made. However, there are techniques to acquire source code from compiled code. These techniques involve the use of a decompiler. A decompiler attempts to make the instructions from the compiled code and reverse engineer them into a language that humans are able to understand. However, due to the relationship between compiled languages and source code many problems arise making decompilers unreliable. Decompiling is difficult because machine code corresponds to the specific hardware that that code is meant to run on. Machine code is a series of instructions, or mnemonics, that is an instruction is actually a number corresponding to a state implemented in hardware. Therefore, different hardware has different instruction set architectures (ISAs) and need different compilers to function. ISAs can be composed of 20 to nearly 1000 instructions. [13]

Software licensing is a complex topic, one with many subtleties. Sofware and software licenses span a wide range of Intellectual property law. Software can be masked by a compiler making it applicable to a trade secret. Software may be copyrighted. Software may even be patentable. When

one considers all of these factors together, one finds that software licenses can be given two general applications. These are apply two different philosophies to software; one a sharing of knowledge another a hording a knowledge. These two philosophies are known respectively as open and closed source.

3.1 Open Source

Open source software licenses, of which there are far fewer licenses, have different goals entirely. Open source licenses use more of its text giving the user rights, while preserving a few key rights to the owner.

Considering the characteristics of closed and open licenses one can see the different incentives to break each license. There are many incentives to breaking a closed source license agreement. One of the more prevalent is illegal copying of commercial software by end users for consumer use. In the End User License Agreement (EULA) for Microsoft's Windows XP operating system there is a section dedicated to this issue alone. The Microsoft EULA is very specific in how it disallows duplicating any of the software that makes up the content regarded in the license. The license is written in way that maximizes the rights Microsoft retains in the event the license is broken.

When analyzing the enforceability of software licenses one must first consider all the parties involved and which are ones are likely to violate particular license agreements. First there are the software developers or the companies that fund the developers, this is the party that issues or writes the licenses. Second, there are consumers who purchase these licenses; this party is referred to as the end users. Third, there are the business users who purchase the licenses, sometimes in bulk with special license agreements.

With these three parties in mind one can start to analyze the parties who are more prone to violate the software's' license. There are roughly two types of licenses issued by the software developers: opened and closed licenses. Here is where the enforceability of the license in question varies. The interaction between the license type and the user of the license influences state of that individual license and how it is enforced.

The terms of a software licenses dictate the enforceability of the license. Closed source licenses

tend to restrict more than open licenses. Thus, closed source licenses have more terms of the license to enforce. Since there are more terms to enforce, closed licenses already have more enforceability problems to deal with. In addition to this the terms of a closed source license are sometimes very specific and therefore inherently more difficult to enforce. For example, the license for Microsoft's Windows XP license states that a user may not run the software on a machine with more than two processors. [1] This allows concerns to surface. The issue of how a term like this of the license can actually be enforced. There are essentially two ways to prevent this activity that breaks the license, from taking place. The first is to build a prevention mechanism into the software itself. For some cases this can be done, but it is also easily broken by a savvy end user. Breaking this mechanism would also result in the violation of the license, leading back to the original issue of enforcing the license. This leads to the second mechanism for ensuring that the terms of the license are not violated by the end user: a "phone home" technique. This is where the software vendor builds into the software a procedure that, through the Internet, contacts one of the software vendors servers. If the software can measure if the license is being broken and then alert the software vendor to this, then action can be pursued. This method has only seen limited deployment in the software industry, mainly in the specialized console gaming industry.

There are also other concerns when it comes to enforcing a software's' license. In the software field, technology advances at a rate fast enough to expire terms of a license while the software is still in its life cycle. To continue with the two processor example, the terms of that part of the license are now more vague than they were when the software license was written. New processor architectures had re-defined what a processor is. Since the writing of this particular license new processor architectures have resulted in several processors "cores" one chip. An industry wide definition of processor core is still somewhat vague. For instance, a cell processor core and an x86 core are both referred to as cores but the core former is missing several components that make up an independent processor. Regardless, with this technological advancement one can emulate a multi-processor machine with a multicore processor, but whether the latter action is allowed by the license is vague.

Clearly, in this field, writing licenses that specific and voluminous, as closed licenses tend to be,

has many problems in terms of enforceability. There are issues when it comes to ensuring an end user obeys a license and more problems when it comes finding what the license actually permits the user to do. Since license enforcement is not straightforward it may not be in the interest of the software vendors to pursue individual end users who violate their license. The cost can be high compared to benefit from stopping an individual end user from violating the license. This leaves the incentive of the software vendor pursue either businesses who violate software licenses or those who facilitate large numbers of individual end users who violate their licenses.

Since most businesses tend to avoid violating software license agreements, due to the risk of a lawsuit, this leaves only the individual violators of the license and those who facilitate software license violations. Facilitating software license violations is not defined in a straightforward manner, there are many subtleties. The best way to illustrate this is by a comparison of Grokster and Bittorrent. Grokster directly facilitates software license violations by allowing users to search other users' shared material. While not all the shared material is copyrighted content a significant amount of it is. From there, the grokster service provides a method for the copyrighted material to be copied from one user to another. Considering this the supreme court ruled that Grokster directly facilitates copyright violations its users participate in. [2] When considering bittorrent the situation is not so clear. Bittorrent is a technology for sharing large files across a network using techniques that distribute the load across all the users sharing the file. With bittorrent a user does not necessarily, and typical does not, share an entire file with another use, though it is possible. Instead, the file is broken up into small chunks which are then shared, so it is possible for a user to share only a portion of a file. Unlike Grokster, bittorrent does not offer a means of locating content, whether copyrighted or not, to share. This must be done by the user. There is a clear difference between Grokster and Bittorrent protocols in both their function and legality. Facilitation is when a person or group makes copyrighted material available, not when a person or group provides a method sharing.

With this in mind bittorrent facilitators have become distributed, there is no one service providing a means to share copyrighted material. When a closed software vendor wishes to enforce its license agreement it is presented a choice: pursue the facilitators of the copyright infringement,

the infringers themselves, or both. This leaves many parties to pursue, both facilitators and sharers. The challenges of enforcing the software licenses can be seen in the numbers. The Business Software Alliance (BSA), a software trade group that is made up of the largest software vendors in the world, such as Apple, IBM, and Microsoft, conducts investigations into the matters of software piracy. The BSA estimates that 20% of the software in the United States is pirated, the lowest in the world. The BSA estimates that the losses due to software piracy in America alone are up to 8 billion US dollars annually. [3]

The structure of software piracy is both large and decentralized, making this task resource intensive to pursue. There is no singular source of pirated software, pirated software is the result of independent individuals who have a personal, and sometimes a business, interest in making pirated software available to consumers. This presents software vendors with different strategies as how to prevent software piracy. Due to the structure of software piracy software vendors shy away from pursuing legal action against pirates or those who use pirated software, though this is not true in all cases. This would almost be frivolous, as new pirates will almost certainly replace those who have been prosecuted. One of techniques software vendors use is copy protection technologies, which make it difficult for software pirates to copy software. Another approach is the software as a service model, which represents a fundamental change in the way software is distributed.

Copy protection software is the older and more prevalent of the two models. It is the more proven of the two techniques, however the software as a service model has much more potential to combat piracy. Although, copy protection technologies have shown some results in combating piracy there are doubts about it's effectiveness in all scenarios. Essentially, copy protection techniques cannot solve this problem entirely, or reduce piracy to low enough levels. There are even question about how much copy protection, if any, if required to minimize software piracy. On one side, too much copy protection, i.e. copy protection that is too aggressive, can harm the manufacturer and the customer. Copy protection software that is over aggressive could be something like a rootkit: where software is unknowingly installed on a legitimate users' machine in order to prevent unauthorized copying. On the other side, there are models that suggest that no copy protection software could be best solution. [4]

Copy protection systems can be successful if applied properly to a given situation. In may cases copy protection systems are circumvented and become only nuisance for software consumers. However, circumventing new copy protection systems can take time, which in some cases is what software vendors need. In the PC games industry many sales are made in the first week of the games release. If the game vendor can apply copy protection that will take a long time to circumvent, then sales can be preserved. However, this implicates that the legitimately purchased product is inferior to the pirated product – since the copy protection has be removed on the pirated product.

Copy protection techniques are not a complete solution to the software piracy problem these methods cannot ensure that all software licenses are not violated. A new model, called Software as a Service, shows promise in helping ensure that software licenses can be adequately enforced. Software as a Service is a technique where the software a company sells is kept on a company server and a user can pay a fee to access the software. For example, Google Apps, which takes advantage of the software as a service model, is a set of programs accessible via the Internet. The user accesses the programs through a web browser and has no access to the source code or the compiled code. Despite this, the Google Terms of Use for Google Apps forbid copying either the source code or the compiled code.[5] Using this model Google can enforce the Terms of Use, that is the license, and be reasonably sure that there are no pirated copies of Google Apps available. There is also another advantage for companies that take advantage of the Software as a Service model. These firms have complete ownership of the code. They do not have to release the source code or the compiled code. Indeed, this is what the authors of Google Apps' Terms of Service agreement included. The Terms of Service state that Google retains all ownership of all right, title and interest of the code that makes up Google Apps. In a traditional model, a software vendor would have to release the compiled code to the customer since the code would be operated on the customers machine. Essentially, the Software as a Service model allows a software vendors to run all of its code on the vendors machine and allow customers to use the software remotely.

Software as a Service is a potential solution to the problem of software license enforcement. Using the Software as a service model firms can write licensees knowing that the license can be enforced properly. However, there are several concerns with the Software as a service model. The

first concern is that of privacy. For example, documents created using Google Apps stay on the server. Although, in the Terms of Service agreement Google specifically forfeits ownership of the created work Google still has the information stored on its servers. Several privacy concerns arise: Google's servers could be compromised, Google employees have access to the servers, and Google uses the data to build profiles of its customers. These profiles are then used to deliver targeted advertisements to its users. Second, a user would require an Internet connection to use any software that is implemented under the Software as a Service model. This represents a large change from previous models, where software is loaded onto the users' device anywhere at any time of the users choosing. Third, this makes the user dependent on the firm providing the service. If, for example, the firms' servers go down the user is completely out of luck, the user would not be able to access or edit their documents. Finally, many questions remain about whether Software as a Service models will become popular. Consumers may not be willing to adapt to subscribing to use software. The current model is that of a one-time fee to use the software. The Software as a service model requires the user to subscribe to the software, or to pay a fee at some time interval, monthly, annually, etc.

The Software as a Service model clearly has many benefits for software vendors, but it is also not a cure-all for enforcing licenses in the software industry. Just like the Software as a Service model open licensing models have their own set of problems to overcome. Enforcing open software licenses differs from enforcing closed licenses in a variety of ways. When open software licenses are enforced the enforcing party is not necessarily the owner or writer of the software lawsuits against license violations can be, and are, filed by anyone. This includes activists and non-profit organizations who write and maintain open licenses. The parties who typically violate open licenses are usually different from those who violate closed licenses as well. Since open licenses tend to focus on user freedom and giving the user and creator of the software rights, as opposed to taking them away, it is less likely for a user to violate an open license.

End users do not make up the party that tends to violate open licenses this is when an open license is violated it usually at the will of corporations. This tendency has many ramifications. The first is that, since corporations have developed legal departments and so many assets that they are afraid to lose corporations are more calculated than end users. Couple this with the small

amount of corporations relative to end users and one will find that open licenses are violated far less frequently than closed licenses.

The text of open licenses is used mostly to give the end user as many rights as possible, while preserving a few basic rights to the software's creator. For example, a contributor to a software project must credit any previous contributors in next version. This is usually not the cause of open license violations. When open licenses are violated, such as the General Public License (GPL) or the Lesser General Public License (LGPL), problems occur when a contributor does not make their contributions open source, as the license demands. This is the situation with an ongoing lawsuit between the Free Software Foundation Inc. (FSF) and Cisco Systems Inc. (Cisco). [6]

In a complaint filed by the FSF, the alleged violations are described. Cisco allegedly made alterations to software licensed under the GPL and LGPL without making the alteration open source, i.e. Cisco did not release the source code to general public under the identical license of the specific software. The difficulty in this case does not lie in volume of those who must be prosecuted, like is the situation for closed licenses. Here, the difficulty lies in proving that Cisco did indeed use source code licensed under GPL. This predicament relates back to one of the fundamental problems regarding intellectual property and software. The software Cisco released is compiled source code, meaning that it releases only the machine readable code to the public. This makes revealing violators of open source licenses difficult. However, if an open source license is uncovered prosecution can be straightforward. For example, a subpoena could be issued that seeks the source code, the source code could then be compared to copyrighted code. This is indeed the approach the FSF has taken in their suit against Cisco. The FSF has requested a subpoena to view the source code Cisco used in several of their networking products.

There are only few examples to show how open source license violations occur. Another example took place in 2007, when a large firm sponsored by the Motion Picture Association of America (MPAA) violated the GPL license. Here, the situation is similar to the that of the previous case, except that no lawsuit was filed. The MPAA had released a toolkit to universities, where the MPAA believes most copyright violations occur, that allowed the university information technology (IT) staff to track and prevent these violations. However, the toolkit released made the use of the GNU

Linux operating system Ubuntu, as well as other open source packages. Ubuntu and the other open source packages were, and still are, licensed under the GPL. The copyright violation was largely the same in this case: a firm used open source software without releasing the modifications. The violation in this case was handled differently. Canonical, the firm that produces Ubuntu issued a DMCA take down notice to the Internet Service Provider (ISP) hosting the toolkit. In section 512 of the DMCA states that if one reports a copyright violation to, in this case the ISP, the ISP is responsible for removing the copyrighted content from the Internet. Whether the a copyright violation has occurred or not, the ISP must still remove the material, the alleged violator can file a lawsuit to keep the content available. This case shows that on occasion open source license violations can be simple to solve. Since the MPAA was attempting to distribute software that was clearly under an open source license, but still forbidding access to the modified source code, the violation was easy to find and then remove.

Software copyright enforcement is a complex issue, due to the nature of software. There are few other fields where one can regard their pre-released (source code) product as both copyrighted material and a trade secret. This affects both closed source and open source licensing differently, but the basis of the two problems are similar. Violations occur when material that is meant to be kept secret. Businesses copyright software to protect their revenue, but then businesses can violate copyrights enhance their revenue. Both of these actions necessitate keeping their source code secret.

Licenses have a large effect on how software is used. Because almost all software is licensed, the license is considered legally binding even if the user does not agree to it. This means that the license holder has the freedom to demand basically whatever they want from the user. This brings us to the issue of licenses affecting sales. A license has a huge affect on a product's popularity. An overly restrictive license will often dissuade your customers from buying the product. (9) Also the license determines whether the product can be given away. An inferior product can often gain significant market share because it is free. Keep in mind there are many different factors that affect the popularity of any software product including: marketing, age, and quality some of which may be more important in determining a products popularity, therefore there are no actual statistics on the effects of licensing.(10)

Code 'ownership' is complicated at the best of times and it becomes more complicated with open source software. Code ownership is governed by the license it is created under. Usually in open source the original creator will be attributed as the owner of the base code and anyone who modifies the code will be attributed as 'owning' the changes they make. You can see how attribution becomes complicated as a project grows. (11) However not all licenses require attribution to be given (keep in mind these are rare). If you were so inclined, you could create a license that forced contributors to turn in their code anonymously, however I don't think anyone has actually created such a license (yet). The question can be examined with a case study of the Linux Kernel. Originally written by Linus Torvalds in 1991 the Linux kernel has had thousands of contributors, both individuals, small companies, large corporations, and individuals with sponsors. The Linux kernel is licensed under GPL. With all these contributors who owns the Linux kernel, ownership of works in question is not mentioned in the GPL. If there is no owner who is liable for any damages the software may cause.

Understanding this project also requires a large amount of technical knowledge of software. At the core of this project lies the issue of source code and compiled code. Source code is the human readable code that a programmer writes. Source code is fed into a compiler and is changed or compiled into compiled code, or machine code that the computer can now run. This is code that the computer can read but humans cannot. Once the code is compiled it is difficult to impossible to translate back to source code, which means that finished programs can be sold to the user without the user being able to tell how they are made. However, there are techniques to acquire source code from compiled code. These techniques involve the use of a decompiler. A decompiler attempts to make the instructions from the compiled code and reverse engineer them into a language that humans are able to understand. However, due to the relationship between compiled languages and source code many problems arise making decompilers unreliable. Decompiling is difficult because machine code corresponds to the specific hardware that that code is meant to run on. Machine code is a series of instructions, or mnemonics, that is an instruction is actually a number corresponding to a state implemented in hardware. Therefore, different hardware have different instruction set architectures (ISAs) and need different compilers to function. ISAs can be composed of 20 to nearly 1000 instructions.(12)

4 Software License Enforceability

The license chosen by a software creator significantly affects the software throughout its lifetime. Therefore, the license that is either chosen for the software in question or written specifically for the software is crucial to whether the software can achieve its initial goals. These initial goals are set by the creator or the party that funds the creation of the software. These goals can include creating monetary wealth for the creating party, creating social wealth for a larger community, and in some cases, creating notoriety for the creator. This last case can be seen in the creation of computer viruses, trojans, etc, where the writer of the software wishes to emphasize his software writing skills to the public. No matter the goal, the license chosen, and ensuring that the software's' users adhere to the terms of the license are crucial to determining how that goal is achieved.

The issue of ensuring that software users follows the license agreement is as old as the software itself. Before advanced communications systems, like the Internet, breeches of software licenses were not a significant issue to the software writing industry. This is due to the change in the medium by which information is spread. Before this change, software licenses were breached by individuals or by word of mouth. Now that it is a trivial task to communicate with any number of people, the volume of software license violations have become impossible to ignore. Knowing this, it has become paramount to the software industry to ensure that these licenses are adhered to. This does not only include commercial software vendors however, the open source community must have the same level assurance that the licenses they assign to their software are upheld

Considering that software license enforcement is important to both open source and commercial software creators one must realize that enforcing these each of these licenses has multiple and varying subtleties. This is a result of the different goals of the license and the goals of the software creator. Closed source licenses must act to protect the business of the firm which created the software. Open licenses must protect the original creator while allowing community members to edit the original software. These two different goals of the licenses affect drastically how the license is to be enforced.

Closed source software vendors have the goal to maximize sales of their software and their licenses are designed to do this. There are two actions that closed source software vendors take to preserve their sales. The first is to protect their source code. Keeping their source code secret not

only prevents software users from compiling the code themselves, but also keeps any innovations secret from any competitors. A closed source license forces the user to agree to, at the minimum, these policies.

This is a legal practice due to the nature of copyright law. Copyrighted material and material that is under trade secret are not necessarily mutually exclusive. Software vendors may gain a federal copyright without submitting their entire body of source code, a vendor only has to submit some of the source to obtain the copyright. This allows software vendors to apply trade secret and copyright status to the material.

Open source software licenses, of which there are far fewer licenses, have different goals entirely. Open source licenses use more of its text to give the user rights, while preserving a few key rights to the owner.

Considering the characteristics of closed and open licenses one can see the different incentives to break each license. There are many incentives to breaking a closed source license agreement. One of the more prevalent is illegal copying of commercial software by end users for consumer use. In the End User License Agreement (EULA) for Microsoft's Windows XP operating system there is a section dedicated to this issue alone. The Microsoft EULA is very specific in how it disallows duplicating any of the software that makes up the content regarded in the license. The license is written in way that maximizes the rights Microsoft retains in the event the license is broken.

When analyzing the enforceability of software licenses one must first consider all the parties involved and which are ones are likely to violate particular license agreements. First there are the software developers or the companies that fund the developers, this is the party that issues or writes the licenses. Second, there are consumers who purchase these licenses; this party is referred to as the end users. Third, there are the business users who purchase the licenses, sometimes in bulk with special license agreements.

With these three parties in mind one can start to analyze the parties who are more prone to violate the software's' license. There are roughly two types of licenses issued by the software developers: opened and closed licenses. Here is where the enforceability of the license in question varies. The interaction between the license type and the user of the license influences state of that

individual license and how it is enforced.

The terms of a software licenses dictate the enforceability of the license. Closed source licenses tend to restrict more than open licenses. Thus, closed source licenses have more terms of the license to enforce. Since there are more terms to enforce, closed licenses already have more enforceability problems to deal with. In addition to this the terms of a closed source license are sometimes very specific and therefore inherently more difficult to enforce. For example, the license for Microsoft's Windows XP license states that a user may not run the software on a machine with more than two processors. [1] This allows concerns to surface. The issue of how a term like this of the license can actually be enforced is not easily discerned. There are essentially two ways to prevent this activity that breaks the license, from taking place. The first is to build a prevention mechanism into the software itself. For some cases this can be done, but it is also easily broken by a savvy end user. Breaking this mechanism would also result in the violation of the license, leading back to the original issue of enforcing the license. This leads to the second mechanism for ensuring that the terms of the license are not violated by the end user: a "phone home" technique. This is where the software vendor builds into the software a procedure that, through the Internet, contacts one of the software vendor's servers. If the software can measure if the license is being broken and then alert the software vendor to this, then action can be pursued. This method has only seen limited deployment in the software industry, mainly in the specialized console gaming industry.

There are also other concerns when it comes to enforcing a software's' license. In the software field, technology advances at a rate fast enough for the terms of the license to expire while the software is still in its life cycle. To continue with the two processor example, the terms of that part of the license are now vaguer than they were when the software license was written. New processor architectures had re-defined what a processor is. Since the writing of this particular license, new processor architectures have resulted in several processor "cores" on one chip. An industry wide definition of processor core is still somewhat vague. For instance, a cell processor core and an x86 core are both referred to as cores but the former core is missing several components that make up an independent processor. Regardless, with this technological advancement one can emulate a multi-processor machine with a multi-core processor, but whether the latter action is allowed by

the license is vague.

Clearly, in this field, writing licenses that specific and voluminous, as closed licenses tend to be, has many problems in terms of enforceability. There are issues when it comes to ensuring an end user obeys a license and more problems when it comes finding what the license actually permits the user to do. Since license enforcement is not straightforward it may not be in the interest of the software vendors to pursue individual end users who violate their license. The cost can be high compared to the benefit from stopping an individual end user from violating the license. This leaves the incentive of the software vendor to pursue either businesses who violate software licenses or those who facilitate large numbers of individual end users who violate their licenses.

4.1 Closed Source

Since most businesses tend to avoid violating software license agreements, due to the risk of a lawsuit, this leaves only the individual violators of the license and those who facilitate software license violations. Facilitating software license violations is not defined in a straightforward manner, there are many subtleties. The best way to illustrate this is by a comparison of Grokster and Bit Torrent. Grokster directly facilitated software license violations by allowing users to search other users' shared material. While not all the shared material is copyrighted content a significant amount of it is. From there, the Grokster service provides a method for the copyrighted material to be copied from one user to another. Considering this the Supreme Court ruled that Grokster directly facilitated copyright violations its users participate in. [2] When considering Bit Torrent the situation is not so clear. Bit Torrent is a technology for sharing large files across a network using techniques that distribute the load across all the users sharing the file. With Bit Torrent a user does not necessarily, and typical does not, share an entire file with another use, though it is possible. Instead, the file is broken up into small chunks which are then shared, so it is possible for a user to share only a portion of a file. Unlike Grokster, Bit Torrent does not offer a means of locating content, whether copyrighted or not, to share. This must be done by the user. There is a clear difference between Grokster and Bit Torrent protocols in both their function and legality. Facilitation is when a person or group makes copyrighted material available, not when a person or

group provides a method for sharing.

With this in mind Bit Torrent facilitators have become distributed; there is no one service providing a means to share copyrighted material. When a closed software vendor wishes to enforce its license agreement it is presented a choice: pursue the facilitators of the copyright infringement, the infringers themselves, or both. This leaves many parties to pursue, both facilitators and sharers. The challenges of enforcing the software licenses can be seen in the numbers. The Business Software Alliance (BSA), a software trade group that is made up of the largest software vendors in the world, such as Apple, IBM, and Microsoft, conducts investigations into the matters of software piracy. The BSA estimates that 20% of the software in the United States is pirated, the lowest in the world. The BSA estimates that the losses due to software piracy in America alone are up to 8 billion US dollars annually. [3]

The structure of software piracy is both large and decentralized, making this task resource intensive to pursue. There is no singular source of pirated software; pirated software is the result of independent individuals who have a personal, and sometimes a business, interest in making pirated software available to consumers. This presents software vendors with different strategies as how to prevent software piracy. Due to the structure of software piracy software vendors shy away from pursuing legal action against pirates or those who use pirated software, though this is not true in all cases. This would almost be frivolous, as new pirates will almost certainly replace those who have been prosecuted. One of the more successful strategies employed by software companies recently has been, rather than suing each infringer individually, companies send a letter demanding payment for the license that the company knows each individual has pirated. If the individual pays, the company receives payment for the software (usually marked up as a penalty for piracy). The much smaller amount of individuals that did not pay will then be prosecuted, making enforcement more manageable.

One of techniques software vendors use is copy protection technologies, which make it difficult for software pirates to copy software. Another approach is the software as a service model, which represents a fundamental change in the way software is distributed.

Copy protection software is the older and more prevalent of the two models. It is the more proven

of the two techniques; however the software as a service model has much more potential to combat piracy. Although, copy protection technologies have shown some results in combating piracy there are doubts about its effectiveness in all scenarios. Essentially, copy protection techniques cannot solve this problem entirely, or reduce piracy to low enough levels. There are even questions about how much copy protection, if any, is required to minimize software piracy. On one side, too much copy protection, i.e. copy protection that is too aggressive, can harm the manufacturer and the customer. Copy protection software that is over aggressive could be something like a root kit: where software is unknowingly installed on a legitimate users' machine in order to prevent unauthorized copying. On the other side, there are models that suggest that no copy protection software could be best solution. [4]

Copy protection systems can be successful if applied properly to a given situation. In many cases copy protection systems are circumvented and become only nuisance for software consumers. However, circumventing new copy protection systems can take time, which in some cases is what software vendors need. In the PC games industry many sales are made in the first week of the games release. If the game vendor can apply copy protection that will take a long time to circumvent, then sales can be preserved. However, this implicates that the legitimately purchased product is inferior to the pirated product – since the copy protection has be removed on the pirated product. The cost of the copy protection system will remain long after the copy protection ceases to be useful.

Copy protection techniques are not a complete solution to the software piracy problem these methods cannot ensure that all software licenses are not violated. A new model, called Software as a Service, shows promise in helping ensure that software licenses can be adequately enforced. Software as a Service is a technique where the software a company sells is kept on a company server and a user can pay a fee to access the software. For example, Google Apps, which takes advantage of the software as a service model, is a set of programs accessible via the Internet. The user accesses the programs through a web browser and has no access to the source code or the compiled code. Despite this, the Google Terms of Use for Google Apps forbid copying either the source code or the compiled code.[5] Using this model Google can enforce the Terms of Use, that is the license, and be reasonably sure that there are no pirated copies of Google Apps available.

There is also another advantage for companies that take advantage of the Software as a Service model. These firms have complete ownership of the code. They do not have to release the source code or the compiled code. Indeed, this is what the authors of Google Apps' Terms of Service agreement included. The Terms of Service state that Google retains all ownership of all right, title and interest of the code that makes up Google Apps. In a traditional model, a software vendor would have to release the compiled code to the customer since the code would be operated on the customers machine. Essentially, the Software as a Service model allows a software vendors to run all of its code on the vendors machine and allow customers to use the software remotely.

Software as a Service is a potential solution to the problem of software license enforcement. Using the Software as a service model firms can write licensees knowing that the license can be enforced properly. However, there are several concerns with the Software as a service model. The first concern is that of privacy. For example, documents created using Google Apps stay on the server. Although, in the Terms of Service agreement Google specifically forfeits ownership of the created work; Google still has the information stored on its servers. Several privacy concerns arise: Google's servers could be compromised, Google employees have access to the servers, and Google uses the data to build profiles of its customers. These profiles are then used to deliver targeted advertisements to its users. Second, a user would require an Internet connection to use any software that is implemented under the Software as a Service model. This represents a large change from previous models, where software is loaded onto the users' device anywhere at any time of the users choosing. Third, this makes the user dependent on the firm providing the service. If, for example, the firms' servers go down the user is completely out of luck, the user would not be able to access or edit their documents. Finally, many questions remain about whether Software as a Service models will become popular. Consumers may not be willing to adapt to subscribing to use software. The current model is that of a onetime fee to use the software. The Software as a service model requires the user to subscribe to the software, or to pay a fee at some time interval, monthly, annually, etc.

4.2 Open Source

The Software as a Service model clearly has many benefits for software vendors, but it is also not a cure-all for enforcing licenses in the software industry. Just like the Software as a Service model open licensing models have their own set of problems to overcome. Enforcing open software licenses differs from enforcing closed licenses in a variety of ways. When open software licenses are enforced the enforcing party is not necessarily the owner or writer of the software lawsuits against license violations can be, and are, filed by anyone. This includes activists and nonprofit organizations who write and maintain open licenses. The parties who typically violate open licenses are usually different from those who violate closed licenses as well. Since open licenses tend to focus on user freedom and giving the user and creator of the software rights, as opposed to taking them away, it is less likely for a user to violate an open license.

End users do not make up the party that tends to violate open licenses. When an open license is violated it usually at the will of corporations. This tendency has many ramifications. The first is that, since corporations have developed legal departments and so many assets that they are afraid to lose, corporations are more calculated than end users. Couple this with the small number of corporations relative to end users and one will find that open licenses are violated far less frequently than closed licenses.

The text of open licenses is used mostly to give the end user as many rights as possible, while preserving a few basic rights to the software's creator. For example, a contributor to a software project must credit any previous contributors in next version. This is usually not the cause of open license violations. When open licenses are violated, such as the General Public License (GPL) or the Lesser General Public License (LGPL), problems occur when a contributor does not make their contributions open source, as the license demands. This is the situation with an ongoing lawsuit between the Free Software Foundation Inc. (FSF) and Cisco Systems Inc. (Cisco). [6]

In a complaint filed by the FSF, the alleged violations are described. Cisco allegedly made alterations to software licensed under the GPL and LGPL without making the alteration open source, i.e. Cisco did not release the source code to general public under the identical license of the specific software. The difficulty in this case does not lie in volume of those who must be prosecuted,

like is the situation for closed licenses. Here, the difficulty lies in proving that Cisco did indeed use source code licensed under GPL. This predicament relates back to one of the fundamental problems regarding intellectual property and software. The software Cisco released is compiled source code, meaning that it releases only the machine readable code to the public. This makes revealing violators of open source licenses difficult. However, if an open source license is uncovered prosecution can be straightforward. For example, a subpoena could be issued that seeks the source code; the source code could then be compared to copyrighted code. This is indeed the approach the FSF has taken in their suit against Cisco. The FSF has requested a subpoena to view the source code Cisco used in several of its networking products.

There are only few examples to show how open source license violations occur. Another example took place in 2007, when a large firm sponsored by the Motion Picture Association of America (MPAA) violated the GPL license. Here, the situation is similar to that of the previous case, except that no lawsuit was filed. The MPAA had released a toolkit to universities, where the MPAA believes most copyright violations occur, that allowed the university information technology (IT) staff to track and prevent these violations. However, the toolkit released made the use of the GNU Linux operating system Ubuntu, as well as other open source packages. Ubuntu and the other open source packages were, and still are, licensed under the GPL. The copyright violation was largely the same in this case: a firm used open source software without releasing the modifications. The violation in this case was handled differently. Canonical, the firm that produces Ubuntu issued a DMCA take down notice to the Internet Service Provider (ISP) hosting the toolkit. Section 512 of the DMCA states that if one reports a copyright violation to, in this case the ISP, the ISP is responsible for removing the copyrighted content from the Internet. Whether a copyright violation has occurred or not, the ISP must still remove the material, the alleged violator can file a lawsuit to keep the content available. This case shows that on occasion open source license violations can be simple to solve. Since the MPAA was attempting to distribute software that was clearly under an open source license, but still forbidding access to the modified source code, the violation was easy to find and then remove.

Software copyright enforcement is a complex issue, due to the nature of software. There are

few other fields where one can regard their pre-released (source code) product as both copyrighted material and a trade secret. This affects both closed source and open source licensing differently, but the basis of the two problems are similar. Violations occur when material that is meant to be kept secret. Businesses place their software under to protect their revenue. This action is necessary to keep their source code secret and thus maintain their revenue.

5 License Restrictiveness and Software Quality

The license that is applied to a certain piece of software affects the use of the software nearly as much as the quality of the software itself. Users choose software not only for what functions the software can perform, but also the license that has been applied to the software. A plethora of issues factor in on the decision of what software to use and how to use software, many of these factors relate to the software's license. The license that a piece of software is licensed under, along with many other factors, can enable widespread use or can render the software usable only to few. Whether or not the software is popular or not depends on the quality of the software and creators intentions: a correctly chosen license can fulfill the creators wishes with regard to popularity. There are instances when the creator of a software package can choose the license to limit the popularity of the software, just like, in more common scenarios, there are occasions when the creator can choose a license to maximize popularity. Correctly choosing a license can also allow the creator to control the methods in the which the software is used. A software creator can influence usage and popularity with both open and closed licenses. Some licenses are defined to allow the author of the software to choose the terms and conditions of the license in an a la carte fashion. There are other factors that impact usage and popularity such as the quality of the software and prior existing standards. For example, Adobe's Portable Document Format (PDF) is a standard and recognized by the International Standards Organization (ISO). [1] Creating a popular alternative to the PDF format would be extremely difficult, no matter the choice of license. Essentially, the license a software creator chooses affects usability and popularity in a manner that is secondary only to the quality of the software itself.

Software licenses affect many characteristics of the software, both apparent and real. Closed

source software is kept closed source in order to protect a software vendor's investment. The argument is that closed source software must be kept closed source so the ideas realized in the software are kept both secret and copyrighted. When software is both secret and copyrighted the protection on the software vendor's investment is maximized. With the investment maximized, the vendor is the unique supplier of this software. This uniqueness can be a feature, an algorithm, or a separate genre of software entirely, what is essential is that the vendor is a unique supplier. When a vendor becomes a unique supplier the user is forced to buy that software and only that software if this uniqueness is desired. The purchase begins the cycle of further investment, as well as profit, for the software vendor. In addition to this model there are other reasons to keep software closed source. For example, the Federal Bureau of Investigations (FBI) keeps some of its software closed source under the guise of national security.[2] This reason, as well as all the other reasons not involving profit seeking are an insignificant minority. So, software that is closed source is generally kept that way to ensure that quality of the software increases. This suggests that there is a relationship between software quality and the terms of the license. This argument suggests that closed licenses produce superior software.

There are also sufficient arguments on the other end of the software licensing spectrum, open source licensing. Open source can, result in software this is high quality. Software that is licensed under an open source license must first be articulated into two distinct categories: commercial open source software and open source software that is the result of individual interest. Commercial open source firms have the same incentive as closed source firms in terms of profit. There are many different subtly different models for selling commercial open source software. For example, SUSE Linux Enterprise Server (SLES) the user pays for a subscription to software updates. Other models include a one-time fee for the software with no updates. In both cases the source code is available to the user in case the user wishes to modify the software. A consequence of individual interest open source business models is more diverse that those of closed source models. These open source firms that release the source code have the added benefit of the open source community. Releasing source code to everyone results in an enormous advantage over closed source software. A curious individual may experiment with open software and find a problem or bug and report to the

software vendor. Instances of this, from perspective of the software vendor, are like a free source code review. The open source community represents a large advantage open source firms have over closed source firms in producing quality software. That is, open source software can be viewed and edited by so many developers that the code can become more refined, i.e. has fewer errors or become more efficient. Open source software has reaped significant benefits from the creation of the Internet. With the Internet any developer can view any open source code project. This allows developers with different backgrounds and coding styles to work together to create better software. Before the Internet, this was not possible. A developer working on a project without the Internet only had colleagues in the immediate vicinity to view and edit source code.

Open and closed source software have different strengths in different markets, in some markets closed source software dominates, while in other open source software is the standard. There are many reasons for this disparity, not all are related to the license that is applied to the software, but in many cases the license influences, directly or indirectly, how widespread the use of a certain software is.

5.1 Closed Source

Microsoft, the world's second largest software company, [3] is the one of the largest advocates of closed source software. Microsoft is assuredly a closed source software vendor that has experienced an enormous amount of success. The market Microsoft dominates most thoroughly is the desktop OS market, Microsoft OS products make up 92% of this market.[3] However, Microsoft also has a large presence in the server OS market, though it's share is not as large. Microsoft's enormous success allows it another critical advantage over open source software vendors when considering market share. Microsoft has the resources to target open source software by manipulating public opinion. This is exactly the practice Microsoft participates in, an active campaign to convince the public that open source software is dangerous for business use.[4] Since the license that is applied to Microsoft products stipulates that Microsoft protect its investment Microsoft has every incentive to promote this agenda. If alternatives to the Microsoft operating system become popular then Microsoft will not be able to invest adequately in its next product, resulting in software of lower

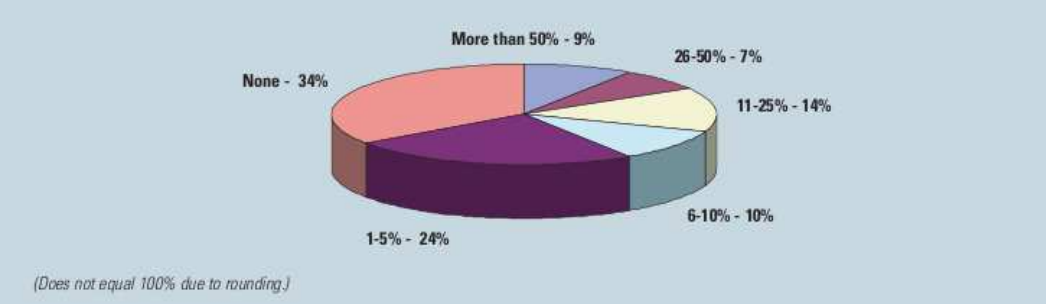
quality. The license that Microsoft applies to its operating system software dictates that Microsoft push this agenda. It is the license that forces Microsoft to engage in the cycle of monetary re-investment of the Microsoft operating system, while keeping the source code secret. If not for such a restrictive license Microsoft would not be under the pressure to maximize profit.

Microsoft's business practices, with regards to its licenses contribute to Microsoft's success in another way as well. Due to this significant reinvestment in its software, the software can be made easier to use. There are tradeoffs in software design: ease of use can be traded off with regard to functionality. That is, software that is easy to use may not necessarily to as powerful as software where there is a large learning curve. This tradeoff between ease of use and functionality can best be illustrated in a comparison between the Microsoft Windows operating system and the GNU Linux operating system. In the Windows environment, nearly everything about the system that can be customized is done so in a graphical user interface (GUI). GUIs are limited by the buttons and lists the software designer chooses to include in the GUI, however they are still simple to use. A user can manipulate settings with only a few clicks of the mouse. On the opposite end is the GNU Linux operating system, where the system is manipulated using the shell. The shell is a prompt where the user types in commands that manipulate the system. Since the prompt is extremely open ended, there are no restrictions on what may be typed in, the system that is being the degree that a system may be customized is much higher. However, due the same openness there is a much larger learning curve that is not all the commands are intuitive. The commands that make up the shell are specialized programming languages, such as the Bourne Again Shell, the korn shell, or even the C shell. In the case of the C shell, C is not even a specialized language, it is very general purpose.

The tradeoffs between ease of use and functionality are a direct result of the license each software distribution has chosen. Due to the pressures of maximizing the reinvestment for its operating system, Microsoft must appeal to the broadest audience possible. In order to appeal to the broadest audience the software must be as simple and intuitive to use as possible. GNU Linux pursues a strategy that fits its needs in a similar way by concentrating not on satisfying the most users, but by satisfying the people who are most active users. For instance, since GNU Linux is

an open source project anyone may contribute, because of this an individual who desires a certain feature may introduce it to the project himself. However, not just any user may contribute; the contributing user must have the relevant technical skills in order to assist in the development of any useful functionality. Again, these two strategies for development are a consequence of the license that is applied to each project, one approach designs for ease of use, the other for functionality.

These two distinct approaches to software design affect how and where the software is applied. As stated prior, Microsoft dominates the desktop user market and this is because this is most general market, the software must appeal to as many people as possible. In markets where specialization is required, for example software applications for business, open source software takes a larger share than close source. As expected, the needs of of businesses are typically greater than that of a general user, requiring software that is more robust. An open source project that typifies popularity at the enterprise level is the Apache hyper-text transfer protocol (HTTP) server. According to a December 2008 Netcraft survey, with a sample size greater than two million websites, the Apache HTTP server held the highest market share with ninety-five million websites making use of the Apache HTTP server. [5] The next most significant entry in the survey was Microsoft's competing HTTP server, with sixty-three million servers using the software.



Apache Web Server Usage [5]

5.2 Open Source

Open source software has such a staggering advantage over closed source software in this field for that very reason: it is open. The most critical characteristics of business class systems are that they are secure and reliable. Open source software has the best prospects to guarantee both security and reliability as a result of being open source, since any interested individual can view the source code

to an open source project. Any developer with an Internet connection can access an open project's source code to contribute, or check the project for bugs or security flaws. Contributions from interested individuals to an open source project are real, and sometimes can be more significant than the contribution of a career developer.

In the summer of 2008 an interested individual uncovered a flaw in an Internet protocol, called the Domain Name System (DNS), that if exploited would have allowed an unscrupulous individual to redirect a user to their own address. For example, if a user put in a request to access a website or an email server a hacker could respond with a substitute of his own website or email server and the user would not know the difference.[2] The founder of the flaw, Dan Kaminsky, discovered the flaw on his own time without any sponsorship, and was able to do so because many, though not all, implementations of DNS are open source software. To clarify, DNS is an Internet protocol, there many vendors that produce software, both open and closed source software, that can perform the functions required of DNS. Kaminsky is precisely the type of individual who helps open source software be as secure and reliable as it is, Kaminsky has expertise, he works for an Internet security firm, that does not consult in the DNS field.

A license does indeed affect how and how often software is used. The license affects how widely adapted the software is. Closed source software, due it's cycle of investment and re-investment, is easier to use making it the more widely adopted type of software. For open source software the tendency is to be more difficult to use, but at the same time present a more powerful set of options to the user. The results of these tendencies are that closed source software is used by users who need software to perform a few basic functions, and open source software is used by users who demand more from their software. Since, the former group represents a larger market than the latter, closed source software has resulted in larger use.

6 Licenses and Software Usage

The software license affects nearly every aspect of the software distributed. The license affects not only how many people use the software, but also the capacity that for which the software is used. The license dictates exactly how the software may be used, the license informs the user

of any limitations involved with copying the software, redistributing the software, adding and subtracting functionality of the software, reselling of the software, and even how many users may use the software at any one given time. From the initial sale or distribution it is the license that is at the center of the software's life span. Given this, software licenses are again divided into two categories, one where the software license is open and the software license is closed. These two different licensing styles, each individual license having subtle differences within its own style, determine everything about how the software is used, apart from the quality of the software itself.

When using software the user has two fundamental choices: open or closed license software. In the case of closed software there are two different choices to be made; they are to pirate or to purchase. There are various factors that are involved in making the choice of whether to purchase or pirate. While purchasing the user must pay for the software, which depending on the software's application can be substantial. There are many benefits to legal acquisition of software; there is technical support, a reduction in liability, and an investment in the software's future versions. The other option, piracy, has its own set of advantages and disadvantages. The first and most significant trade-off to pirating software is the lack of any cost at all, but the result of that lack of cost is an increase in liability – pirating software is indeed illegal. There are also small disadvantages; like that the technical support is not available to software pirates, as well as any software updates. When a user makes the decision to pirate software, the user essentially decide whether they are more interested in minimizing cost or minimizing liability.

The trade-off between cost and liability affects different groups of software users differently. This follows from different groups having different amounts of money to designate on software expenditures and, in many cases, have fewer liabilities. A group of software users, for example a business that has the funds to spend on industrial software packages, also, since it has the funds to afford the software in the first place, has assets that make pirating software a poor business decision. That is, the risks of pirating software, then having a lawsuit filed against the firm makes the option of pirating software actually more expensive, on average, than just purchasing the software. In the long run, a well run firm, as long as that firm has significant assets, will choose the option of purchasing software over pirating software because it is cheaper. To firms with large assets, pirating

software is not worth the risk.

There is however, an alternative group to a business where piracy is a much more economically viable option. This group lies in the individual consumer. To the consumer, pirating software then provides a path. On average, that path is less expensive than purchasing the software legally. The typical situation of the individual consumer provides an environment where piracy can thrive for two key reasons. The first is that individual consumers have fewer assets; the individual consumer has far less to lose than a business. The second is scale, when an individual consumer pirates one software package that event can easily go unnoticed by the bodies that enforce software copyright infringement. However, if a business were to pirate software, the scale of that piracy would have to be large. If a business participated in large scale software piracy, there would be many avenues for the business to be caught and prosecuted, ranging from righteous employees to noticeable internet traffic of pirated software. Essentially, the incentive to pirate software lies with consumers and the incentive to purchase software lies with business.

The attractiveness of engaging in software piracy changes when one considers others costs to the user as well. There are many more subtle costs to software piracy, such as circumventing the copy protection and find a source to pirate the software from. Add this to the to the costs associated with all software, whether pirated or not, such as learning how to use the software, one can find the total cost of any software package. When the pirating software the costs are: copy protection, find a source to pirate the software from, maintaining the software, learning the software, and the taking on the risk of a lawsuit. When purchasing software the costs are: the price the software is sold at, the cost to learn the software, and the cost of maintaining the software.

In both cases, the cost of learning how to use the software is exactly the same, pirating or buying software cannot reduce the learning curve. However, some costs do vary with the method by which the software was obtained. Software maintenance is an example of this. In the case of pirating the software, there is only one option to maintain the software: pirate the newest version and start over. So, for every version update the copy protection must circumvented again and it may not even by the same of copy protection. There are two sub-cases for legitimately purchased software. One is that the software offers free updates and the update process is straight forward,

the Microsoft Windows update style is an example of this. Alternatively, the software updates may need to be purchased; the Mathworks' Matlab software package uses this model.

Clearly, the decision to pirate can be complex and different for every software package depending on the software maintenance structure. However, this added complexity mainly affects the consumer, as the business that pirates software is rare. For business, the risks of pirating software are just too large to engage in pirating software. This is even truer for larger business that have even more to lose and more methods of being caught.

Open source licensed software presents an alternative to pirating software. That is, the risks associated with pirated software are not present in any capacity with open licensed software. Like pirated software open source software is free to download from the channel, depending on the software. The difference being that acquiring open source software is legal and straightforward, where downloading pirated software is certainly illegal and is not necessarily straightforward. Like pirated software, there is no cost involved with acquiring open source software, by definition it is free. However, there may be other costs, as mentioned prior, involved with open source software. These costs, including learning how to use the software, are not monetary but must be considered by the user none-the-less. As discussed earlier open source software typically involves a large commitment to learn how to use the software.

The learning curve associated with open source software is not just limited to open source software. Open source software may require more commitment to learn, but there are many different communities around every major open source project, as well as plenty of documentation in most mature projects. And, as mentioned earlier, some open source firms make it their business model to sell technical support. Essentially, if a user needs assistance with an open source project, help can be found through a variety of channels. This is also true legitimately for purchased closed source software, though the methods of providing assistance may be different, it can still be found. However, for pirated software finding help is not easy, if possible at all. When pirating software, burden of technical support and learning is on the user. This lack of help can be magnified by the extra knowledge that can be required to use pirated software. This is, for example, copy protection circumvention. There are no firms that will provide support to a user looking to break a copy

protection scheme. There is also the additional learning curve that is associated with any software. This phenomenon helps to narrow the field of potential pirates down to only the experienced or technologically apt user. This, ironically, is the group of users that makes up the open source community.

Open source software however, is not without any problems of its own. The problems with some open source software licenses is that, for some licenses, the terms of the license dictate the license that created works must be licensed under. For most open source projects, such as those licensed under the GPL, this does not apply. For example, if an open source project already exists and a developer adds on to the project, then those additions must be made under that same license, according to the GPL. However, there are a small number of infectious licenses that dictate the license that any newly created works must be licensed under. For example, a developer can create a new project and use software to aid the developer in the development process. If however, the aiding software's license stipulates that any software created from the aiding software's use must have the same the license, then the developer loses rights. These rights are essentially the right to their own intellectual property. [1]

The license that is applied to a software package, has a tremendous effect on that software. Some results of the license obvious, like code ownership or the right to copy. However, some are very subtle and are the result of the needs of the people who use software the most. Open source software has a different audience and therefore its users have different needs, likewise for closed source software. It is the license that allows a software package to appeal to different audiences.

7 Source Code Ownership

Ownership is a complex issue in Open Source Software from a legal standpoint. However from a practical standpoint it can often be much simpler.

7.1 Closed Source Ownership

In traditional closed source software, when a software project is written by one programmer, then the copyright belongs solely to that single programmer. He or she may use, control, or distribute

that software as he or she wishes (within reason). When project is completed by multiple programmers and their work is not completed under a contractual agreement, then the copyright of the resulting software is shared between the contributors. The question of whether a programmer owns a part copyright to the entire piece of software, the entire copyright to the part of the software that he or she has worked on or altered, or to some combination of the two has not been legally defined and the amount that an individual has contributed in a closed source project is often impossible to tell in practice, especially when trade secrets are involved. For this reason, almost all software companies make programmers sign a contract that stipulates that the programmer gives ownership of the software to the company. The software is commissioned on behalf of the company and then licensed to the user.

7.2 Open Source Ownership

The difference between closed and open source licenses is that open source licenses require that the released software be open and editable by users. Because attribution is often important to getting contributors, individual contributors and the changes they add are usually tracked carefully. Open source licenses enforce the concept that programmers have copyright and ownership only for the code that they write or the changes that they make. This implies that code ownership is not shared, but rather that individual contributors own 'parts' of the software, rather than the kind of shared copyright that would result from for instance a literary work. However unlike in a closed source license, owners have fewer rights to how the software is used and distributed. The GPL, the most popular open license, ensures that derivative works must be released under a similar license and users are not required to accept the license unless they alter the software. Also bug fix submitters or other non code submitters often must give up any rights for finding the bug in return for attribution. This depends on the license the submitter submits under.

Other licenses like the BSD license allow a user or company to release previously open software as a closed source derivative. This means that even though the original contributors still own the copyright to the part of the software that they developed, the individual or company that releases the derivative work has the right to release the software under any license that they choose.

When a company, individual, or community is developing a software project, at any time another individual, community, or company can 'fork' an Open Source Project, or make a copy of the original and begin developing a project in a new direction. While this does mean that anyone can create a competing product from the code that another company has already written, it does not mean that a company will have the money, recognition, or most importantly community established to support it. In practice most Open Source projects are not forked except in the case of a project being abandoned or mismanaged, where instead a community tends to gather as a group to support the project more efficiently.

There are of course exceptions, Like the Open Source Operating System Linux, where the high interest level and market share allows for multiple communities with different business models. Even in this case, it is usually more cost effective to contribute to an open source effort than to create a competing Open Source Project, because the end result is often the same.

All of this means that in practice, ownership of the copyright in Open Source Software is usually a much less important issue. Most often, whoever leads an open source community controls how the code is developed. There is often a large difference in the amount of control the leadership of an open source project has or uses, because projects are sometimes managed by Corporations or companies, by a small group of developers, or by a committee of developers in a community. The leadership of an open source project can also have a large amount of influence on what the community submits for patches or bug fixes, how a project is marketed, and other aspects of a community. In addition, the leadership of a community decides how donation money is spent. The money could be used to pay developers or contributors and encourage further contribution, or more likely it could be put toward site and bandwidth costs, toward marketing the project, and toward other costs.

7.3 Business Licenses and Open Source Ownership

Companies like IBM and Microsoft often create special licenses when writing open source software. These licenses tend to have special clauses indemnifying the creating company and protecting the companies patents as well as its copyrights. For all other practical purposes, they tend to act like

the FreeBSD license (for instance, Microsoft's Public License) or the GNU license (for instance, Microsoft's Reciprocal License) [1]. They require the code that was written by the parent company to be given attribution and some licenses even require any new code to be compiled in a separate folder (for instance, in some parts of IBM's Lotus Symphony License [2]), so the ownership of the already written code is very clear. These are the kind of restrictions that businesses that develop Open Source may sometimes place on their software. While compiling separately is not always practical, depending on circumstances and the language the code is written in, it can be used to further preserve a companys status as owner of the source code.

8 Digital Rights Management as a Response to Piracy

Piracy has had a large effect on the Software Industry. Its effect is both highly touted and impossible to measure accurately. The Business Software Alliance has been doing studies on the amount of piracy since 2001.[1] However this leaves 10 years unmeasured before the Internet became popularly used and 20 years unmeasured since the creation of the personal computer. [2] Although we will refer to BSA studies for numbers, one should also keep in mind that these are estimates and other organizations estimate different numbers. Piracy is not a recent development, it has been around as long as copyrighted software has been around, as evidenced by Bill Gates' open letter to Hobbyists.[3] In the letter, Gates asserts that most computer hobbyist steal software. Bill Gates wrote the letter in 1976 showing that by that point in time, even though piracy mostly only involved sharing floppy disks in person, piracy was already entrenched problem to the industry. (For the record, Microsoft was created in 1978)

8.1 The effects of Digital rights Management

The industry created many Digital Rights Management technologies to prevent piracy over the years. The need for DRM can be outlined by the letter that Gates wrote. Users were using Altair BASIC but not paying for it. Gates was losing money selling software to hobbyists. Digital Rights Management was the solution that allows software publishers to restrict the users from selling the software, the idea being that users would be forced to buy the software instead. There were many

DRM schemes that were created that are no longer used because they are too costly, complicated, or too inconvenient to the user. We will not examine those. We will also not bother with DRM schemes that are not widely distributed or those made for media other than software. I will also note that there is a large amount of money that gets spent on creating these DRM 'solutions'. Although the exact amount would be hard to measure, large companies like Microsoft, Apple, and EA are spending large amounts of money and becoming increasingly invested in DRM. [4] The problem is that in practice, all DRM schemes are impossible to enforce. The reason is a basic tenant of encryption theory. In order to allow the user to use the software, you must provide the user with all the ingredients to break the DRM. [4] This is the reason that for all DRM schemes faced with a large enough user base, the DRM is broken, usually within days. [4]

The earliest anti-piracy measures like CD checks and serial numbers were simple and mostly ineffective, however they still tend to be the most widely used anti-piracy measures, because they are simple, relatively unobtrusive compared to other DRM systems developed, and cheap to implement. They also allow for a license in which the software, once bought, belongs to the end user so that they may hold property rights over it like selling it or saving a copy. The idea with these technologies is to stop piracy from the most unsophisticated users while not inconveniencing all their legitimate users too much. CD checks and serial number checks are meant to prevent copied programs from working. Although legally anyone can copy software and own as much as they would like, some licenses may restrict that. (We will discuss the legality of that later.) Historically these DRM schemes were only effective because users had to share physical copies of the software.

As the Internet grew in popularity, DRM become increasingly ineffective as the ability to share 'cracked' files became increasingly easy. As a result, more aggressive DRM was developed. Technologies like StarForce were developed that only run on certain hardware and prevent DRM encrypted files from running while you run certain other software. StarForce and SecuRom in particular has received criticism for being intrusive and inconvenient to users, as well as making some legally purchased software stop working, and being hard to remove. [5]

The Internet has allowed for a different kind of DRM in the form of services like Steam. [6] These services allow a user to download a piece of software, which will 'call home' in order to ensure

that the software is not pirated. This scheme relies on the company selling the software to maintain its servers so that users may use the software. This allows for a license in which the company owns the software and allows the user to use the software, rather than giving ownership of the program to the user. This is something implied in many software licenses over time and is an important legal precedent. There is an advantage to the users in this scheme, in that users no longer has to maintain the software themselves so so users no longer have to worry about backing up that piece of software. Taking this DRM scheme to an extreme would resemble the DRM scheme in SPORE in which the user may only install the software on a certain number of machines. [7] This is closer to the digital equivalent of renting the software. It should be mentioned that neither of these DRM schemes actually prevents piracy from happening; the DRM schemes are still breakable. It is also worth mentioning that outrage over SPOREs DRM scheme caused it to be one of the most pirated games ever.[7]

Currently, one of the most difficult types of encryption to break is called trusted computing. [8] The idea behind trusted computing is that the computer hardware would only run a trusted operating system and the operating system would only run trusted programs which are authorized by a remote server. So if a program is encrypted to require trust then the program will have the operating system dial into the remote server, which then decides whether the computer has trust (presumably, the user has paid for the software.) This method is especially hard to break because trust is not defined by the user. However it is still possible to overcome this DRM measure. [9] This is type of encryption is coupled with a software license in which the software is licensed, not sold, to the user. Additionally it can be licensed under an agreement where the game is rented to the user because the software needs to dial into the remote server. Once again, the disadvantage to the user of this method is that when the server is down the user looses access to the software that they purchased. If the server is down permanently, for instance, in the case the company goes out of business, then the users are cut off from their software permanently.

8.2 Software as a Service

The only truly impossible to break digital rights management method is to not release your software in the first place. The next best thing however, is to not release the important part. The software as a service model is relatively simple. The user gets a client piece of software that does nothing except communicate with the server side software. The important data is kept on the server, as well as the useful parts of the software you plan to sell. Because the useful part of the program is kept on a remote server, and because the user must be authorized to use the server in order to connect to the software (presumably, the user paid), the software cannot be used without permission. An alternate form of this scheme is to do as much work as you can on the client side and simply do all the calculation on the server side that would be impossible to do on a single client computer. An extra advantage of this setup is that even if your server side software is stolen, it is economically unfeasible for a user to use the software. At that point, it is much cheaper to buy your software then to rent an expensive server. An example of a program run like this would be World of Warcraft. Keep in mind this scheme is most practical only when it is necessary, for instance, large multi-user programs require servers, so users expect the need to connect to them. However most single user programs do not require a server to run, which makes this setup especially inconvenient to users. In this setup, the emphasis shifts from worries about cracking and piracy to worries about reverse engineering, account hacking, and employees leaking the software. Unfortunately, it is sometimes possible to reverse engineer server side software. For instance, there is now software that emulates World of Warcraft. [10] Keep in mind all these DRM schemes result in development costs and any server requirements involve additional costs.

8.3 Reducing Piracy

One of the reasons that including DRM in software products has not been a large factor in customer purchasing habits is because historically most customers were ignorant of the effects of Digital Rights Management. However, considering the recent customer boycott of Electronic Arts products [11] it becomes increasingly important to handle DRM responsibly. EA broke the boycott of their products by moving their software to the service Steam. [12] There is now a general industry

acceptance that making DRM too aggressive is bad for business. By DRMs nature, the more effective it is, the more inconvenient it is to the consumer. Therefore logically the optimal amount of DRM from the customers' point of view is usually no DRM. Because all DRM is breakable, DRM provides inconvenience to legitimate customers, but not pirates. This encourages piracy. Cutting out DRM can also save money. [13]

Note that CD checks, hardware level checks, and dial home checks are only effective in preventing users from backing up data and in preventing person to person sharing. Any cracked version of the program can be freely shared. Dial home checks are effective for stopping programs that require a server. For example, for the game Halo, the single player part of the game will work fine if it is cracked, but the multiplayer part of the program will not function without a valid serial number. For this reason, many software companies are trying to adopt software as a service model. For some software, the software as a service model makes great sense, because some programs require a server in order to work. However, most software programs do not require a server. Considering the increased cost associated with maintaining servers, the added inconvenience to the user, as well as the questionable gain from preventing piracy, it is usually best to avoid using it if possible.

Some of the popular justifications for including DRM in software products are that DRM is meant to keep honest users honest, that DRM is only meant to stop the most unsophisticated users, and that some companies make the largest amount of money at the opening launch of the software. The problem with the argument that DRM keeps honest users honest is that there are legitimate uses that an honest user may want to do with the product. Making the product difficult to crack encourages users to use a pirated copy. DRM only stops the most unsophisticated users from sharing files, but the global nature of peer to peer networks means that the amount individual users can share is negligible compared to networks of people. The claim that DRM needs only to last a short time in for instance, the video game industry, where 90% of the sales of most video games happens in the first two week is harder to refute. However empirical evidence shows that removing DRM from games does not cause the piracy rate to increase.[14] This mean that there is no evidence that DRM reduces piracy. Unfortunately we lack enough data that could tell either way. Keep in mind that even if a software company removes DRM from later versions of their

product, that means they've still inconvenienced at least 90% of their customers.

The good news is that there are many ways to lower the amount of piracy that are not DRM, or even use piracy to increase sales. If 90% of a companies products are pirated, then by getting only another 10% of the pirates to convert into customers, the companies sales double. The most important thing a software company can do is provide a better product than pirates can provide, or at least one that is not worse than the freely available pirated alternative. This can be as simple removing DRM and other roadblocks that prevent your customers from using their products, to more advanced systems that allow customers to use software after it has only been partway downloaded, to one of the most effective systems of piracy prevention, product updates. [15] For instance, Microsoft provides its genuine advantage check, which allows customers to get free updates to Windows, but only for paying customers. There are many ways to handle product updates, but the important idea is that they make a legitimate copy of the software much more convenient and valuable than a pirated copy.

To quote Kevin Kelly, co-founder of Wired Magazine, "When copies are free, you need to sell things which cannot be copied" [16] Product updates make the software more convent to use than the pirated version. Customers are also often willing to pay to get the software sooner, so allowing customers to get software products before and faster than pirates is important. Customers are often willing to pay for personalized software that would not be available pirated. For instance, a business may need software developed for their company or a gamer may be willing to pay more for a game box signed by the developers. Another way to make customers more willing to buy your products is to sell service. For instance, the open source Operating System Red Hat is distributed freely but customers have to pay from technical support. Finally one of the best ways for a company to lower piracy of its products is to get customers to like their products as well as their company. While this is a marketing effort, putting forth a positive image as a company scales well. Once a company convinces users to be fans of their products, then users will want to pay for products even if they can get them for free.[16] Creating a community for users of a companys software and encouraging developers to take part in that community is a good business decision for many software companies.

9 Software Piracy Rates

9.1 Piracy rates by country and Industry

One of the important things to recognize about software piracy is that not all software is pirated equally. Take a look at the numbers. The Business Software Alliance estimates piracy rates at 21% in North America and 38% Worldwide [1], meaning that 2 out of 5 of all software products in the world are pirated. The piracy rate is 82% in China, which is understandable when you take into account the average salary of people in China. Most Chinese cannot afford the cost of software priced for American consumers and businesses at this time. This brings us to a way that developers can benefit from piracy. [2] Microsoft has acknowledged that allowing for piracy allows companies to compete with free and open source alternatives, as well as establish themselves inside weaker economies. Mainly piracy allows Microsoft products like Vista to become the industry standard, even in regions like China, where Vista is unaffordable. This makes it more likely that the users will buy a legitimate copy once Vista is affordable. Apply this same idea to users who may be too young or too poor to buy the current version of your software and you see the reason why piracy is not considered as large a problem as it could be.[2]

Different industries inside the software world are affected differently by piracy and should treat their anti-piracy measures differently. [3] For instance: Excluding video games, the estimates of the share of pirated software is 34% for Productivity software, 23% for Operating System software, 8% for Document Management software, and 35% for all other types of software. Video games are in a different category because the piracy rate is so high. The current estimate is that 90% of all computer video games are pirated. That's 90% worldwide and around 70% in North America. [4] [5] Obviously piracy is amplified in the video games industry. In one study it was found that for every 1000 illegal downloads of the game Ricochet Infinity that software developer Reflexive could prevent, 1 additional sale would be gained. This reinforces the perception that piracy is often a crime of convenience. [6]

Software Type	Share of Pirated Titles
Productivity	34%
Operating Systems	23%
Document Management	8%
Creative	7%
Security	6%
Utilities	6%
Development	4%
Database	3%
Media Management	3%
Server	2%
CAD	1%
Accounting	1%
Mapping	1%

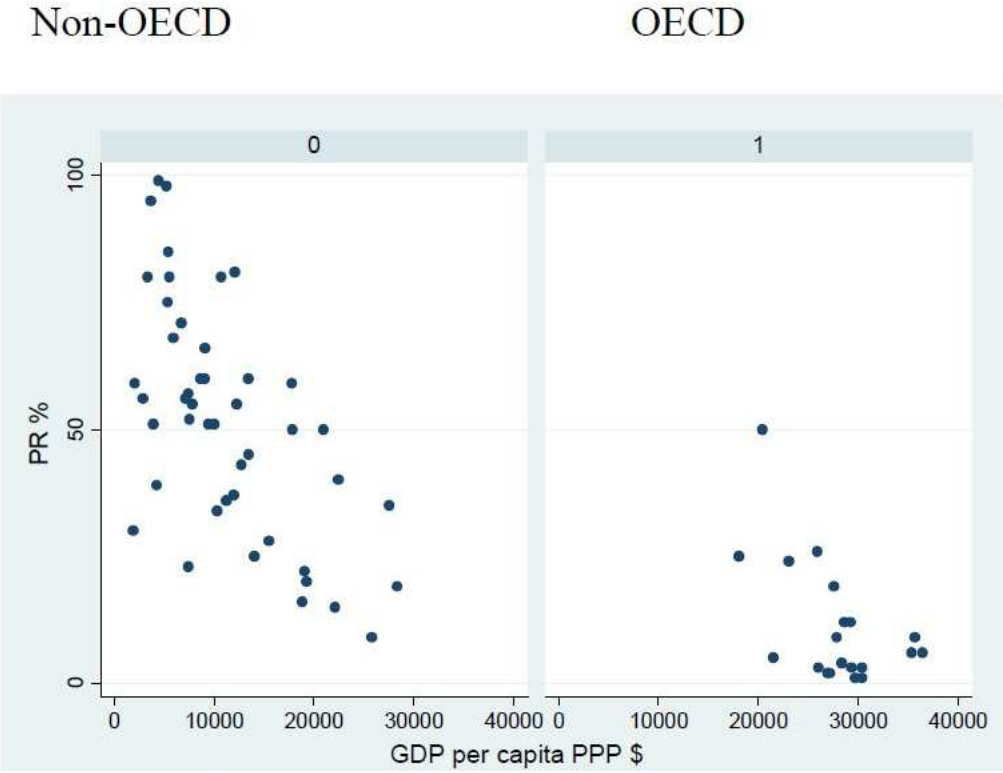
Types of Pirated Software Reported in 2007
Credit: Report on Operations SIIA Anti-Piracy 2007 Year in Review, SIIA

The question of why some software is pirated more than others is complicated, but by far the most important factor is the target audience of the software. Because piracy rates are different for different industries, the amount that a company's software is pirated depends on the type of software that it is. For instance, accounting software accounts for 1% of pirated software while productivity software accounts for 34% percent of pirated software. [3] Although more productivity software is sold than accounting software, there are not 34 times as much productivity software sold as accounting software. This means that a larger proportion of users of productivity software pirate software compared to accounting software.

Because these figures are generalized over two different software industries, it is unlikely that the reason for the drastically different piracy rate is due to any merit or flaw of the companies selling the software or the software itself. The difference between the two industries piracy rates is more likely due to their differing user bases. Coordinating Gross Domestic Product per capita data with piracy rate data reveals some completely unsurprising data .[7] Countries where citizens make less money pirate more software.

While the piracy rate never reaches 0% this means we can reasonably assume that users are more likely to buy software, if they make more money and are better able to pay for it. This is why the target audience of a software product is important in determining how much it is pirated. This means products that are designed to appeal to a large audience regardless of their ability to

pay are more likely to be pirated than products that are designed for users or businesses with a higher income level. Businesses are also much more likely to buy software rather than pirate it, to protect themselves from legal action. Note that even if software is pirated more, that does not mean that it is less profitable.



GDP per Capita v. Piracy Rates

Credit: IFPI and World Bank 2004

The amount an individual software product is pirated is, of course, influenced by its popularity. For instance, even though CAD software took up only 1% of the share of pirated titles in 2007, Autodesk AutoCAD was the fifth top infringed title in 2007. An interesting effect of piracy is that sometimes users will pirate programs that they would not pay for if they could. This effect shows up often in computer video games where users may consider a game worth their time but not their money. This means that high quality as well as mediocre quality will sometimes raise piracy rates. The amount of piracy is also affected by the price of the software. Again the ability for a large amount of users to pay for software is limited by their economic status. There are large number

factors, like DRM, that affect the piracy rate. It is often hard to tell the effects of the many factors involved in the piracy rate, even if we know they are important.

9.2 Licensing and Piracy Rates

Another important part in determining whether a product is pirated is the license it is released under, because of course, redistributing an open source product is not piracy. Releasing an open source version of a software product in parallel to a closed source product has been a successful strategy in reducing piracy by giving pirates a free and legal alternative to pirating a product. The software company SugarCRM has had success by releasing a community edition of their product for free with about 85% functionality of their Professional Edition, which they sell licenses for.

In terms of EULAs, most individual users are not concerned with license agreements. Studies show [8] that only a small percentage of users read licenses, so for most consumers the contents of the license is unimportant. However unreasonable licenses have occasionally been the cause of lawsuits [9] and more often a cause of bad press. For example, when Google released its browser Chrome, it garnered negative press over a badly worded part of the license which gave Google the rights to anything written in the browser. Google changed the license after receiving complaints, but it demonstrates the importance of careful licensing. [10] Depending on the size of a company, licensing terms become more important as larger companies are under more pressure to follow the licenses. In addition large companies sometimes negotiate license terms, so creating an enforceable useful license even more important when dealing with businesses than individual users.

While the problem of piracy is a significant one for the software industry, it not by any means a death sentence. The computer software industry as a whole has experienced steady growth from 2001 to 2007.[11] This means that it is possible to succeed in spite of piracy and hopefully use it to your advantage. Digital Rights Management was created in response to piracy and a need to compete with free and open source products. Some consumers moved to open source products in response to DRM making the software inferior. In the end even making a convenient, high quality product isn't always enough to succeed. In a real market success is governed by quality, marketing, timing, luck, and a host of other factors. What we can recommend are some methods to improve

the quality of a software product, increase its convenience, attractiveness to consumers, and/or make it harder to pirate.

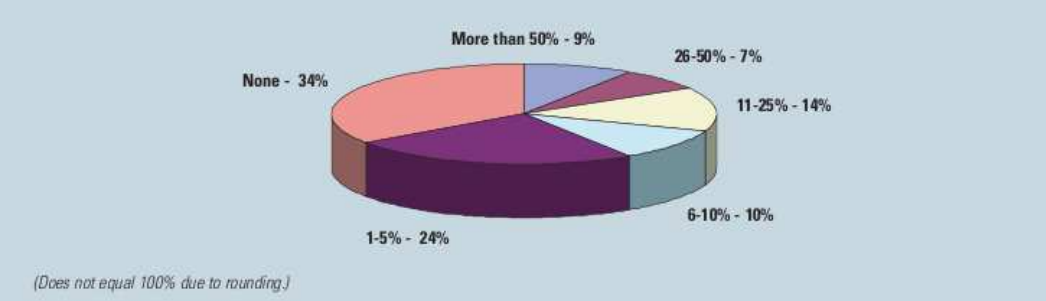
10 Open Source and Business

When developing software it is important to know what audience that the software is being developed for. Here we look at what affects the type of software that companies use and the difference in software use between companies and individuals. Open source software has not been universally embraced or accepted in the business world. Some companies actively push for its adoption and some companies resist using open source. There are various arguments for and against open source software, some of which are valid. There are various conflicting reports on the adoption of open source software into the market so there are no real numbers. [1][2] However most reports agree that the number is higher than 50% in the US and even higher in some developing economies where users are required by cost constraints to not use commercial software. [3] The reason for this discrepancy is explained by an Information Week interview with Yahoo Developer Jeremy Zawodny.[4] In the interview Zawodny is quoted saying: "Developers have quite a bit of leeway when finding the best tools to get the job done," so often even if a company does not have a formal policy for or against Open Source software, it's employees may be using it. BSA reports also claim that usage of pirated software is the same for businesses and individuals, but this is unlikely. [5]

10.1 What Businesses use Software

Because it is often hard to fit a business model to open source in some industries or types of software development, and because the amount of contribution to an open source project is often dependent on how exciting or important it is considered, there are large gaps in the availability and quality of different types of open source software depending on the type of software. Because of that and the fact that open source software may have additional cost to deploy or use depending on the type of software, the adoption of open source software is different depending on the industry. [6] [7] Also because of the exponential nature of the number of computers and quantity of server equipment required for larger companies, the amount of savings for adopting open source software

is larger for larger companies. [8] For this reason Open Source Software is often targeted at large companies, and also at small companies or individual users in order to obtain more contributors. For this reason open source software is adopted by differing amounts and used differently based on a companys size. [9] [10]



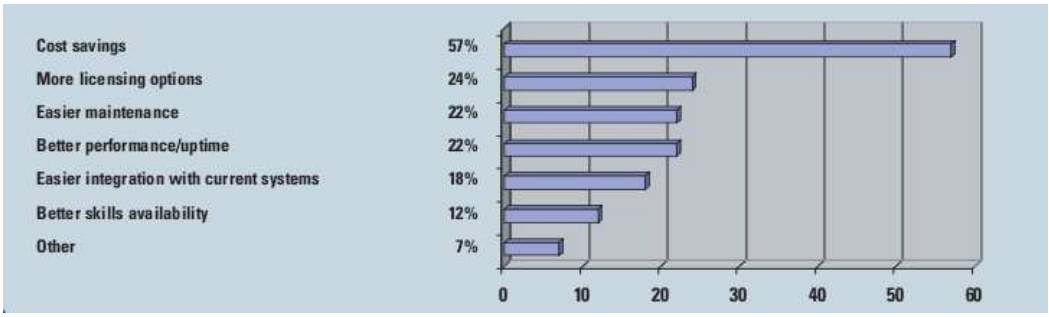
Percent of Mission Critical Operations Running on Open Source [9]

A larger percentage of individual users as a whole tend to use less Open Source Software than the percentage of companies that use open source software because if some people in a company are using open source software, the company as a whole is considered to be using open source software in some surveys. This is why the Open Source Institute was able to claim that 85% of companies use Open Source. However there is some evidence that Open Source software is adopted more highly in businesses. For instance, even though open source browsers only make up around 30% market share [11] and Linux operating systems make up around 0.36% of the global market share (and 8% of internet connected PCs) [12], the share of open source servers is higher than the number of closed source servers. Apache alone commands approximately 60% of the market share for web servers. [13]

10.2 Where is Open Source Successful

There are good reasons to choose either an open or closed source program based on the circumstances. One of the often cited reasons for companies to stay away from open source software is the lack of indemnity in open source software. Indemnity is the lack of an agreement in the license that the licenser will pay insurance if the software fails on the end users machine. Open Source software is almost always created 'as is' with a lack of indemnity to encourage contribution

by protecting the individual contributors from legal action. (For instance even the Apache Server operating system is licensed 'as is'.) Often, in open source software legal recourse would also be logistically impossible because of the many unrelated contributors in multiple different countries. This means that with most open source software the responsibility for critical software failures is shifted from the software developer to the user and the user is left with no legal recourse against the developer. In some situations, the benefit of getting indemnity more than offsets the cost of software and can be useful in protecting a company from losses as the result of software that it purchased. While this is usually not important for consumers or small businesses, it is important for large corporations that buy large amounts of software.

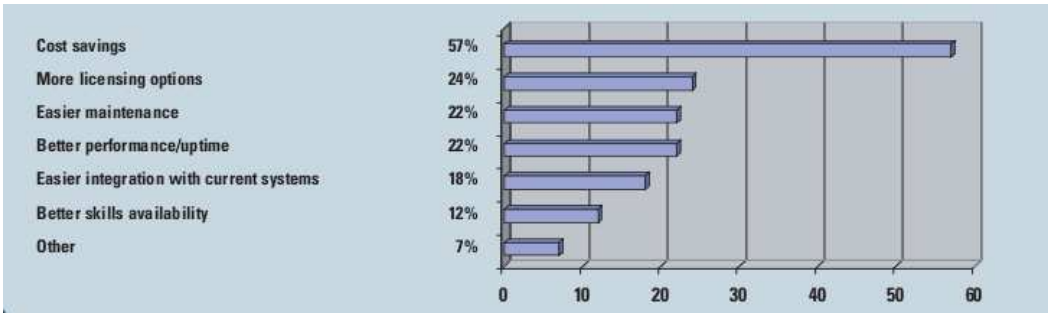


Factors Driving Open Source Adoptation [9]

However, many closed source products also do not have indemnity protection. If indemnity is required for a company's software, the company often has to shop around or undergo costly legal negotiations among closed source vendors. On the other hand choosing open source software can often protect end users from legal action. [10] This is because unless a company alters the software, open source software does not require users to accept a license. Therefore the user is not bound by potentially unreasonable demands or legal costs, and is not required to keep track of licensing paperwork. The BSA and other similar organizations seek to fight piracy, but it is the responsibility of a company to prove that it did not pirate software. In other words, in a licensing dispute, innocent until proven guilty does not apply, because the users own a license to the software, not a copy of the software. [14] Once again, the license that a piece of software is distributed under is very important. We interviewed a software developer in a major software company about this subject. We were told that open source software has to be checked by the company's legal team

before employees are allowed to use the software. This lets the company ensure to stockholders that the software is safe and well constructed.

10.3 Commissioned Software and Open Source



Main Limitations of Open Source [9]

The third choice that a company has, other than using an open or closed source product, is to develop the product it needs internally or have it commissioned. This is an expensive prospect and the cost is based on the size of the project. Nevertheless, a large portion of the software market is created by commission and a large amount must be created that way. For instance, embedded software must be commissioned or written by the hardware company. The advantage of this method is that the creating company owns the copyright to the created code. This means that according the company's preference, the software could be kept inside the company, it could be distributed or sold commercially, or it could be released as an open source project. This brings up the question, are companies more likely to use open source software if they develop the software themselves, or if they have a hand in development?

11 Conclusion

Several conclusions can be drawn from the use of open and closed source software licenses in business. Each license type affects every business that relies on software, both directly and indirectly. Closed source license violations are more difficult for software firms to prosecute than open source license violations are, due to the volume of prosecutions that must occur. Closed source violators must be prosecuted individually. On the other hand open source licenses can only be violated by

a party that writes software based on an open source license. This is a much smaller group and often consists of companies and individuals that are easier to prosecute. Additionally, many open source violations occur by accident where the conflict may be resolved without trial.

In order for firms to protect themselves legally, they ensure that there will be no consequences for using open source software. This includes assuring owners that the open source software a firm uses is safe, that is, that the open source software must be trust worthy. Additionally if the company is a software development company, it much choose software tools with licenses that will not interfere. Despite this, most firms do not have a policy regarding the use of open source software. [1] If one of these firms is large, then the likelihood of open source software being used increases. Additionally, if a firm lacks a similar policy regarding pirated software, then its employees are more likely to engage in the use of pirated software.

Ownership of open source code is not straightforward. Since open source projects are open source, any developer can work on the project, and this leads to a large number of contributors. With so many contributors code, ownership is not well defined. However, ownership of an open source project would have little value. In open source projects, it is often less important to own the intellectual property to the written code, because the distributed nature of open source code allows for many developers making small contributions. Who owns a specific piece of code is less important because anyone can copy, redistribute or edit that piece of code, even when multiple developers or companies have altered it.

To contrast, in situations where closed source licenses are used, the concept of ownership is defined more clearly. Since closed source licenses are sold for profit, the owner must be definite, that is, some individual or firm must be acquiring those profits. Because developing a business model for open source is difficult, closed source and open source often succeed in different areas. Open source tends to be more functional and closed source software tends to be more user friendly. This is because most open source developers develop software for themselves. OSS tends to be more developer oriented, while closed source software is written mostly for people who could not develop software for themselves.

The large volume of owners, or developers, is a strength of open source. Since there are a

large number of developers, some may find problems that others may not. Also, the large number of developers increases the probability of locating insertions of malicious code or security holes. Where open source benefits from a larger pool of developers, closed source benefits from a cycle of investment and reinvestment. Because of this, it is easier to develop a closed source business model than an open source business model. We recommend that open source business models need to develop further. Donations, merchandising, and technical support based models are all complementary business models. To thrive on a larger scale, open source software firms must adapt to different business models.

Software piracy is a phenomenon that affects both open and closed source licensing. Piracy rates vary widely when considering different software. Open source software, by definition, cannot be pirated. On the other hand, the amount that closed source software is pirated varies greatly. Some closed source software is hardly pirated at all, while other software packages suffer from 90% piracy rates. This is related, foremost, to the size of the market the software is developed for. Larger markets, which are mostly made up of individual consumers, tend to pirate the most. Smaller markets, for specialized software that is more typically useful for a business are less likely to be pirated. Consumers also tend to have less to lose from a lawsuit than a business and are much harder to catch.

Americans have one of the highest per capita incomes in the world and America is one of the largest countries in the world. This makes America the largest market for computer software and for this reason most commercial software is targeted at and priced for Americans. American consumers pirate the least software of any country because American consumers as a whole can afford the most software. The rate of piracy in a country is inversely proportional to the countrys average income. However after a countrys income reaches a certain point piracy tends to level off. As a country becomes more developed its piracy rate decreases dramatically, while economic changes in the developed world have much less of an effect.

Digital Rights Management is one proposed solution to reducing piracy. Unfortunately, not only does it almost always fail to work, it also has a definite cost involved. DRM systems have to be maintained and they inconvenience customers. The more powerful the DRM system is, the

greater the inconvenience to customers. The more aggressive the DRM system is, the more users that will choose not to buy the software. This is why the optimal amount of DRM to include in software is no DRM. The question then becomes how to deal with piracy without DRM. Ironically, the best solution to piracy is to take advantage of it.

Piracy allows consumers who would not have been able to afford the software to use the software, and it allows users who would not have bought the software in the first place to use the software. There is usually a percentage of pirates that will have only have bought a piece of software if it was unavailable to pirate. Luckily, this market is actually considered quite small. The Video Games industry has the highest piracy rate of any software industry. Studies have shown that for every 1000 downloads of a game that can be prevented 1 sale is gained. [2][3][4] So, even if the video games industry was pirated 10 times as much as all the other industry (it isn't), then by blocking all piracy sales will only increase by 1%. Preventing piracy is not a sustainable solution, and even if it was, closed source companies now have to compete with free and open source software, which does not require a license or DRM to use. A much better solution is to turn pirates into users.

Pirates are users of a piece of software. Closed source software relies on users paying for software so that more software can be written. Obviously, users of a current version of a piece of software are more likely to pay for the next version of a piece of software. Microsoft has looked the other way to piracy in developing markets like China so that when consumers will be able to afford Windows, Microsoft will already have market saturation. This means piracy can be a great boost to a company's consumer base. The other way to take advantage of piracy is to turn users who would have never bought the software until they tried it, into customers. While software can be pirated, there are many things that cannot be pirated and there are ways to make legitimate, non-pirated, copies of a piece of software better than a pirated version like: automatic updates, building a community between users, and increased functionality for legitimate users who can connect to a remote server. Taking the server side methodology to the extreme is the software as a service model. The software as a service model lets consumers use a server's programs only after they have connected and presumably paid. While it isn't perfect, the software as a service model is often a good choice for serving software and is sometimes the only way to do some types of computing.

Software as service also provides the user with increased flexibility. A user need not be limited to one machine when using web applications; all that is needed is an Internet connected computer.

The future of software licenses will depend on the use of DRM, and its alternatives, by software vendors. Independent of this, open source software's popularity will continue to grow. Most likely, one will never overtake or replace the other, but they will continue to complement each other.

12 Reseach Needs

We have decided to include a section recommending where we think further research could be done and where we found current research lacking. Here, we propose a series of mostly unanswered research questions . There is little research on the history of piracy. How has piracy shaped the software industry historically? We found that there is little research on who actually pirates software and who provides pirated software. Little research was found that provided the motives of the people who provide pirated software. How much do companies pirate software compared to individuals? Quantitative information for piracy rates among businesses and different sizes of businesses is needed. Does the size of the company affect its piracy rate?

There is little research on the people that pirate software. Are pirates mostly those who have low incomes? How much do people in different age groups pirate software? Data related to software pirates who are older than 25 is needed. Why do pirates pirate software? Do pirates have motives other than saving money? How much of software that is pirated would actually be purchased? What are the real losses due to piracy?

There is a great amount of empirical and anecdotal evidence, but there is little solid data on the effects of DRM on consumers. How does DRM affect consumer buying habits and piracy? What is the average technical knowledge of a pirate compared to a user who does not pirate? Is DRM only a formality to software pirates? Does the average user even notice DRM in software? Does it increase piracy? What is the perceived risk of pirating software? Do the perceived risks differ to the software pirate and non-pirate? What is the real risk of pirating software? Does the average software pirate consider the risks involved in pirating software? Do average software users perceive a greater risk than technically advanced users?

There is a great amount of data on businesses and open source, but there are some research gaps. There is also a research gap on the individual users and contributors to open source. What is the average technical knowledge of an open source user? How much software is commissioned or developed internally for companies, rather than sold as commercial software? How much commissioned software is open source? Would there be a benefit to making software open if it is not being sold commercially?

13 Bibliography

Introduction

1- Bruce Perens. (2006).
The Open Source Definition. The Open Source Institute.
Retrieved Jan 6, 2009, from
http://opensource.org/docs/osd

2- Microsoft Corporation. (2009).
Open Source at Microsoft: Licenses. Microsoft Corporation community
Retrieved January 15, 2009, from
http://www.microsoft.com/opensource/licenses.mspx

Background

1- Philip H. Albert. (2003).
Free and Open Source Software Interview Questions. FindLaw.
Retrieved January 15, 2009, from
http://library.findlaw.com/2003/Dec/19/133224.html

2- Ryan Paul. (2007).
MPAAs University Toolkit hit with DMCA takedown notice after GPL violation.
Ars Technica
Retrieved January 15, 2009, from
http://arstechnica.com/news.ars/post/20071204-mpaas-university-toolkit-hit-with-dmca-takedown-notice-after-gpl-violation.html

3- Free Software Foundation, Inc.(2007)
GNU GENERAL PUBLIC LICENSE, Free Software Foundation, Inc.
Retrieved January 25, 2009, from
http://www.gnu.org/copyleft/gpl.htmldefault

4- Open Source Initiative (2006)
The BSD License, Open Source Initiative
Retrieved January 25, 2009, from
http://www.opensource.org/licenses/bsd-license.php

5 - Creative Commons (2009)
Creative Commons Licenses, Creative Commons
Retrieved January 25, 2009, from
http://creativecommons.org/licenses/

6 Ryan Paul. (2007).
MPAAs University Toolkit hit with DMCA takedown notice after GPL violation.,
Ars Technica
Retrieved January 15, 2009, from
http://arstechnica.com/news.ars/post/20071204-mpaas-university-
toolkit-hit-with-dmca-takedown-notice-after-gpl-violation.html

7 JMRI. (2008).
Court documents for the JMRI Defense Case., JMRI
Retrieved January 15, 2009, from
http://jmri.sourceforge.net/k/docket/index.shtml

8-Lloyd L. Rich. (1994).
If You Use A Shrinkwrap License It May Not Be Enforceable:
Mass Market Software and The Shrinkwrap License., FindLaw.
Retrieved January 15, 2009, from
http://library.findlaw.com/1994/Jun/1/130938.html

9 -Peter Gutmann (2007)
A Cost Analysis of Windows Vista Content Protection
Retrieved January 17, 2009, from
http://www.cs.auckland.ac.nz/˜pgut001/pubs/vista˙cost.html

10- Various. (2008).
Customer Reviews : Spore. Amazon.
Retrieved January 16, 2009, from
http://www.amazon.com/review/product/B000FKBCX4/ ref=pd˙bbs˙sr˙1˙cm˙cr˙acr˙img?˙encoding=UTF8&s

11 -Kevin Kelly (2008)
Better Than Free, The Technicum.
Retrieved January 16, 2009, from
http://www.kk.org/thetechnium/archives/2008/01/better˙than˙fre.php

12-Mozilla Corporation (2008)
Distributed Decision Making: Mozilla Modules and Module Ownership, Mozilla.org
Retrieved January 16, 2009, from
http://www.mozilla.org/hacking/module-ownership

13-Intel (2008)
Linux Kernel Compiling

Retrieved January 16, 2009, from
http://software.intel.com/en-us/articles/linux-kernel-compiling

Software License Enforceability

1 - Zymaris, C. (2005),
'A Comparison of the GPL and the Microsoft EULA', .

2 - MGM Studios, Inc. v. Grokster, Ltd., 545 U.S. 2780 2005

3 - The Business software Alliance (2007)
BSA/IDC Global Software Piracy Study, BSA
Retrieved January 16, 2009, from
http://www.bsa.org/country/Research and Statistics.aspx

4 - Software Piracy: An Analysis of Protection Strategies
Kathleen Reavis Conner , Richard P. Rumelt

5 - Google (2009)
GOOGLE APPS STANDARD EDITION AGREEMENT, Google
Retrieved January 25, 2009, from
http://www.google.com/apps/intl/en/terms/standard'terms.html

6 Free Software Foundation Inc. v. Cisco Systems Inc., (S. D. NY 2008)

7 - Washington Post, Brian Krebs, 11/23/2007,
MPAA University 'Toolkit' Raises Privacy Concerns

License Restrictiveness and Software Quality

1 - ISO standard 32000-1

2 - Lessig, L. (2006),
Code,
Basic Books,
387 Park Avenue South,
New York, NY 100168810,
pp. 165-172.

4 - Forbes (2007)
Special Report The Global 2000, Forbes
Retrieved January 25, 2009, from
http://www.forbes.com/lists/2007/18/biz˙07forbes2000˙The-Global-2000˙IndName˙17.html

5 - Raymond, E. S. (2001),
'Why Microsoft smears-and fears-open source',
IEEE Spectrum 38(8), 14

6 - WSS (2007)
Market Share for Top Servers Across All Domains, Netcraft
Retrieved February 15, 2009, from
http://news.netcraft.com/archives/2007/04/02/
april˙2007˙web˙server˙survey.html

Licenses and Software Usage

1 Microsoft Corporation. (2009).

Open Source at Microsoft: Licenses. Microsoft Corporation

Retrieved January 15, 2009, from

http://www.microsoft.com/opensource/licenses.mspx

2 IBM (2008)

IBM License Information document: IBM Lotus Symphony 1.0.0, IBM

Retrieved February 2, 2009, from

http://www03.ibm.com/software/sla/sladb.nsf/lilookup/ 03FAFC0D1896CD9B0025745C000F109Fopendoc

Source Code Ownership

1 The Business software Alliance (2007)

BSA/IDC Global Software Piracy Study, BSA

Retrieved January 16, 2009, from

http://www.bsa.org/country/Research and Statistics.aspx

2 Mary Bellis (2008)

Inventors of the Modern Computer, About.com

Retrieved January 16, 2009, from

http://inventors.about.com/library/weekly/aa031599.htm

3 Bill Gates (1976)

Bill Gates' Open Letter to Hobbyists, DigiBarm Computer Museum

Retrieved January 16, 2009, from

http://www.digibarn.com/collections/newsletters/homebrew/

V2˙01/gatesletter.html

4 Cory Doctorow(2004)
Microsoft Research DRM talk.
Retrieved January 16, 2009, from
http://www.craphound.com/msftdrm.txt

5 StarForce (2009)
StarForce Copy Protection System
Retrieved January 16, 2009, from
http://www.star-force.com/

6 Steam (2009)
Steam Client. Valve Corporation
Retrieved January 16, 2009, from
http://store.steampowered.com/about/

7 Various. (2008).
Customer Reviews : Spore. Amazon.
Retrieved January 16, 2009, from
http://www.amazon.com/review/product/B000FKBCX4/ ref=pd˙bbs˙sr˙1˙cm˙cr˙acr˙img?˙encoding=UTF8&s

8 Peter Gutmann (2007)
A Cost Analysis of Windows Vista Content Protection
Retrieved January 17, 2009, from
http://www.cs.auckland.ac.nz/˜pgut001/pubs/vista˙cost.html

9 SECUDE International AG (2008)
Understanding DRAM Attacks, IT SEC SWiSS AG.
Retrieved January 17, 2009, from

http://secude.com/download/htm/10810/en/White˙Paper:˙Don't˙Panic˙-˙Cold˙Boot˙Reality˙Check.pdf

10 Electronic Frontier Foundation (2005)
Blizzard v. BNETD, Electronic Frontier Foundation
Retrieved January 17, 2009, from
http://www.eff.org/cases/blizzard-v-bnetd

11 Ben Kuchera (2008)
The EA boycott, Ars Technica
Retrieved February 13, 2009, from
http://arstechnica.com/news.ars/post/20081015-eas-drm-ceo-arrogance-may-cause-gamers-to-skip-good-titles.html

12 Legendary Studio (2007)
CRYSIS WARHEAD COMING TO STEAM, Legendary Studio
Retrieved Febuary 4, 2009, from
http://www.crytek.com/news/news/? tx˙ttnews[tt˙news]=128&tx˙ttnews[backPid]=1&cHash=6fc3c000da

13 Ken Fisher (2008)
75% of customer service problems caused by DRM, Ars Technica and Musicload
Retrieved February 13, 2009, from
http://arstechnica.com/news.ars/post/20070318-75-percent-customer-problems-caused-by-drm.html

14-2D Boy (2008)
90%, 2D Boy
Retrieved February 13, 2009, from
http://2dboy.com/2008/11/13/90/

15 - Brad Wardell, President/CEO of Stardock (2003)
Piracy & PC Gaming, Opinionated techie
CD Copy protection is not the way to stop piracy, Opinionated techie
Retrieved February 13, 2009, from
http://draginol.joeuser.com/article/209

16 Kevin Kelly (2008)
Better Than Free, The Technicum.
Retrieved January 16, 2009, from
http://www.kk.org/thetechnium/archives/2008/01/better˙than˙fre.php

17 - Steven Frank (2008)
The First, the Free, and The Good,
dead link as of January 16, 2009, from
http://stevenf.com/archive/the-first--the-free--and-the-good.php

Digital Rights Management as a Response to Piracy

1 The Business software Alliance (2007)
BSA/IDC Global Software Piracy Study, BSA
Retrieved January 16, 2009, from
http://www.bsa.org/country/Research and Statistics.aspx

2 Charles Piller (2006)
How Piracy Opens Doors for Windows, Los Angeles Times
Retrieved January 17, 2009, from
http://articles.latimes.com/2006/apr/09/business/fi-micropiracy9

3 SIIA, (2007)

Report on Operations SIIA Anti-Piracy 2007 Year in Review, SIIA

Retrieved February 2, 2009, from

http://www.siia.net/piracy/yir˙2007.pdf

4 Russell Carroll (2008)

Casual Games and Piracy: The Truth, Gamasutra

Retrieved February 13, 2009, from

http://www.gamasutra.com/php-bin/news˙index.php?story=17350

5 2D Boy (2008)

90%, 2D Boy

Retrieved February 13, 2009, from

http://2dboy.com/2008/11/13/90/

6 Brad Wardell, President/CEO of Stardock (2003)

Piracy & PC Gaming, Opinionated techie

CD Copy protection is not the way to stop piracy, Opinionated techie

Retrieved February 13, 2009, from

http://draginol.joeuser.com/article/209

7 Valentina Assenova (2007)

Determinants of the Music Piracy Divide, The Wharton School-University of Pennsylvania

Retrieved February 15, 2009, from

http://titan.iwu.edu/˜econ/uer/articles/UER%202006-2007%20Assenova.pdf

8 Larry Magid (2009)

It Pays To Read License Agreements, PC Pitstop

Retrieved Febuary 4, 2009, from
http://www.pcpitstop.com/spycheck/eula.asp

9 Lisa M. Bowman (2003)
Lawsuit challenges software licensing, CNet
Retrieved February 15, 2009, from
http://news.cnet.com/2100-1001-983988.html

10 Ina Fried (2008)
Google backtracks on Chrome license terms, CNet
Retrieved February 15, 2009, from
http://news.cnet.com/8301-13860˙3-10031703-56.html

11 MSN financial
Retrieved February 13, 2009, from
http://moneycentral.msn.com/home.asp

Software Piracy Rates

1 Mark Alvarez (2008)
85% of Companies Use Open Source Softwares, Study Says, Open Source Initiative
Retrieved February 15, 2009, from
http://www.atelier-us.com/e-business-and-it/article/gartner-85-of-companies-use-open-source

2 Actuate Co. (2007)
More than half of all German companies use Open Source, heise online
Retrieved February 15, 2009, from
http://www.heise.de/english/newsticker/news/91447

3 Jeff Atwood (2008)
We Don't Use Software That Costs Money Here, Coding Horror
Retrieved February 15, 2009, from
http://www.codinghorror.com/blog/archives/001097.html

4 Larry Greenemeier (2005)
Why In The World Would Big Companies Use Open Source?, Information Week
Retrieved February 15, 2009, from
http://www.informationweek.com/blog/main/archives/2005/09/
why˙in˙the˙worl.html

5 The Business software Alliance (2007)
BSA/IDC Global Software Piracy Study, BSA
Retrieved January 16, 2009, from
http://www.bsa.org/country/Research and Statistics.aspx

6 Bull Corporation (2008)
Commissioned Study Shows Open Source Paves The Way For The Next Generation Of
Enterprise IT, Bull Corporation
Retrieved February 15, 2009, from
http://www.wcm.bull.com/internet/pr/rend.jsp?DocId=412289&lang=en

7 Unisys Corporation (2007)
Independently Researched Study for Unisys Shows Growing Acceptance of Open Source
in Business Critical Applications , Unisys Corporation
Retrieved February 15, 2009, from
http://www.businesswire.com/portal/site/google/? ndmViewId=news˙view&newsId=20070502005082&newsL

8 John Blau (2003)

Big companies save big from open source, InfoWorld

Retrieved February 15, 2009, from

http://www.infoworld.com/article/03/05/08/HNbigopensource˙1.html

9 Joe McKendrick (2006)

Survey on Open Source Trends, IOUG

Retrieved February 15, 2009, from

http://oracleweb.ioug.org/Portals/0/researchwire/ResearchWire-OpenSource.pdf

%1 2 4 11

10 David A. Wheeler (2007)

Why Open Source Software? Look at the Numbers!, David A. Wheeler

Retrieved February 15, 2009, from

http://www.dwheeler.com/oss˙fs˙why.html

11 Net Applications (2009)

Browser Version Market Share, Net Applications

Retrieved February 15, 2009, from

http://marketshare.hitslink.com/report.aspx?qprid=2

12 OneStat (2006)

The 10 most popular operating systems in the world, OneStat

Retrieved February 15, 2009, from

http://www.onestat.com/html/aboutus˙pressbox46-operating-systems-market-share.html

13 WSS (2007)

Market Share for Top Servers Across All Domains, Netcraft
Retrieved February 15, 2009, from
http://news.netcraft.com/archives/2007/04/02/
april˙2007˙web˙server˙survey.html

14 Joyce Slaton (2002)
Tangling with the Business Software Alliance can mean big problems, San Fransisco Cronicle
Retrieved February 15, 2009, from
http://www.sfgate.com/cgi-bin/article.cgi?f ile=/gate/archive/2002/02/07/bsa.DTL
Conclusion

1- Mark Alvarez (2008)
85% of Companies Use Open Source Softwares, Study Says, Open Source Initiative
Retrieved February 15, 2009, from
http://www.atelier-us.com/e-business-and-it/article/gartner-85-of-companies-use-open-source

2- SIIA, (2007)
Report on Operations SIIA Anti-Piracy 2007 Year in Review, SIIA
Retrieved February 2, 2009, from
http://www.siia.net/piracy/yir˙2007.pdf

3-Russell Carroll (2008)
Casual Games and Piracy: The Truth, Gamasutra
Retrieved February 13, 2009, from
http://www.gamasutra.com/php-bin/news˙index.php?story=17350

4- 2D Boy (2008)
90%, 2D Boy

Retrieved February 13, 2009, from
http://2dboy.com/2008/11/13/90/

Related Material : (Research done that was not cited in this report.)

(1)npost (2007)
Interview with Don MacAskill, CEO of SmugMug ,npost
Retrieved February 15, 2009, from
http://www.npost.com/interview.jsp?intID=INT00183
http://www3.interscience.wiley.com/journal/118583566/abstract

(2)Shamus Young
Articles Relating to Piracy and DRM, TwentySidedTale
Retrieved February 15, 2009, from
http://www.shamusyoung.com/twentysidedtale/?p=1807
http://www.shamusyoung.com/twentysidedtale/?p=1558

(3) Larry Lessig
Free Culture
http://www.free-culture.cc/

(4) The Economist.com
The benefits of Piracy
Retrieved February 15, 2009, from
http://www.economist.com/opinion/displaystory.cfm?story˙id=11750492

(5) Ryan Paul
An organization that collects music royalties in the UK has sued law enforcement authorities for play-

ing music in police stations without a proper license. No, really. ,Ars Technica
Retrieved February 15, 2009, from
http://arstechnica.com/news.ars/post/20080613-uk-police-nicked-by-copyright-cop-for-playing-radio-too-loud.html

(6)Financial Times
Music industry should embrace illegal websites, Financial Times
Retrieved February 15, 2009, from
http://www.ft.com/cms/s/0/e72884f6-6175-11dd-af94-000077b07658.html?nclick˙check=1

Interviews:

(1) We interviewed an anonymous Developer in a major software vendor over email. This interview was edited to remove any potentially sensitive information and for spelling and grammar.

Question 1: Is [company name removed] interested in any software distribution models besides Software as a Service?

Answer: We make exposure.

Question 2: Does [company name removed] policy or culture encourage developers to use open source software?

Answer 2: Yes we are encouraged to use open source software which has gone through Intellectual Property Laws approval for use. We use only Open Source software which has been reviewed by the law team, and gives us the ability to commit to our customers that we are delivering a good solid product that may not have any IP infringements, also ensures we do not overwhelm the community and ensure the commits back to the community are controlled and good commits, without contaminating the community with bad code, and avoid IP issues.

Question 3: Why are some [company name removed] projects made open, while others are not? Answer 3. Opportunities with Open Source software, limit the amount of time to develop solutions. and enable others to build services and solutions.

Question 4: What is [company name]'s business opportunity in open source software (e.g. [product names]) ?
4. We evaluate software distribution models all the time, SAS, onsite, hosted are all options which [company name] delivers.

Question 5: Do you think that [company name removed] is concerned with some it's closed source software being pirated?
Answer 5: [company name removed] is concerned and actually signed a deal with [different company name removed] licensing and other licensing providers to ensure we are not victims of licensing fraud.

Question 6: [company name removed] has used and even developed Digital Rights Management schemes in the past. How much DRM is included in [company name removed] software right now? Answer 6. There is no DRM in [software project the developer is working on], and I am unsure if there is any in [company name] brands. I think we developed the IP property for DRM and license it to business partners, but we don't have any that I know of.

(2) We interviewed Copyright Lawyer Mary Casey and George France, one of the three founders

of the open source based company handhelds.org. This was a teleconferenced interview so we only have notes rather than a transcript of our conversation.

We were told by Mary Casey that with any type of open source license, the creator of the project has a choice of whatever license they choose. For a license with a 'share alike' clause, such as the GPL, any contributors are required to abide by the license and license their contributions under the same license.

George France said that some licenses are even more strict, requiring all software packaged with the software project or developed using the software project to be licensed under then same license. Most licenses, like the GPL, are specifically designed to avoid this. However if these stricter types of licenses are used with commercial software, then the software must be licensed instead as open source. This why companies must be careful about what licenses are included in software, and the tools that companies use. There are now companies that specialize in analyzing licenses in order to protect companies from these types of license 'infections'.

Under a share alike license, the original creator has the copyright to whatever they write and the derivative creators have copyright to whatever they write themselves. Coders only have ownership to code they write themselves even if their code is used in other projects. Programmers only get copyright to what they create. Copyright and trade secrets do not overlap legally, this is why code can be copyrighted and be a trade secret at the same time.

The 1st circuit court has not handled copyright cases very well. The court is very in consisted and not in line with the rest of the country on copyright law. For this reason, copyright cases are hard top litigate. Often other solutions are in order. The traditional solution to copyright infringement is to send a cease and desist letter followed by litigation if the infringer does not comply. One of the newer, more successful models has been to send a letter requiring the infringer to pay for the license they have infringed upon. If the infringer does not comply then legal action is taken. This allows both parties to avoid costly litigation.