FREE AND OPEN SOURCE SOFTWARE: AN INTRODUCTION

Gwyn Firth Murray±
Matau Legal Group
www.mataulegal.com

with

Michael A. Duncheon±±
Hanson, Bridgett, Marcus, Vlahos & Rudy LLP
www.hansonbridgett.com

January 2006

± Gwyn Firth Murray was one of the "early adopters" in the legal community of a focus on open source software. She is founder and principal of the Matau Legal Group, which offers a broad range of commercial, licensing, and other legal services to both start-up and established companies in the high tech and biotech industries (see www.mataulegal.com). Ms. Murray is co-founder of Open Bar, Inc., a not-for-profit organization focused on legal rights and responsibilities in the world of open source software (see www.open-bar.org) and currently serves as Vice-Chair of the Open Source Committee of the American Bar Association Section of Science and Technology. Ms. Murray is a graduate of Stanford University Law School, and also holds an M.A. in Latin American Studies from Stanford University. She obtained her B.A. magna cum laude and with distinction in economics from Yale College.

±± Michael A. Duncheon, and his firm, Hanson, Bridgett, Marcus, Vlahos & Rudy, LLP, provide a range of legal services to companies and associations in high technology markets, including advice relating to intellectual property, antitrust, standard setting, licensing, and corporate formation. Michael is a graduate of the Stanford Law School (J.D. 1975, Order of the Coif) and Stanford University (A.B. Phi Beta Kappa.)

±±± This paper was produced at the request of and with the financial support of the Storage Networking Industry Association (www.snia.org), which the authors gratefully acknowledge.

FREE AND OPEN SOURCE SOFTWARE: AN INTRODUCTION

Gwyn Firth Murray
Matau Legal Group

with

Michael A. Duncheon
Hanson, Bridgett, Marcus, Vlahos & Rudy LLP

Managers of companies that thus far have been operating exclusively in the proprietary technology space frequently ask us to provide a "quick" explanation of the basics of Free and Open Source Software ("FOSS"). [1] They are hearing more and more about "Free" or "Open Source" software and think it may be time to learn something about it. In particular, these managers increasingly wonder whether they should take a look to see whether they have any open source code inside their company and product code bases. We urge them to do so, knowing that, when they do, they likely will be surprised by the extent to which FOSS has infiltrated their enterprise without their knowing. This is a key point: perhaps the greatest danger that companies that use or rely on technology for their business face today with respect to FOSS is assuming that it isn't there or that it doesn't matter.

The goal of this paper is to help answer the call we have heard for a brief "primer" on the basics of FOSS, including a summary of its history, characteristics and increasing influence. We also aim to clarify how FOSS licensing is both similar to and different from "traditional proprietary" software licensing, and to describe briefly some of the essential licensing schemes within the FOSS Community. Finally, we offer some thoughts on how companies can engage with FOSS without panicking or exposing themselves to undue risk -- and be better positioned to reap its many benefits.

WHAT IS FOSS?

The simplest way to define FOSS is that it is software whose source code is freely available to be used, copied and modified. FOSS software programs can be proprietary or non-proprietary, and can be used for commercial or nonprofit purposes. While this

[1] We refer to the programs discussed in this paper as "FOSS" because, as explained in this paper, the terms "Free Software" and "Open Source Software may be used interchangeably. However, these terms do not necessarily mean the same thing and, as dicussed in this paper, were developed from different perspectives within the FOSS Community. Use of one term to the exclusion of the other does not cover the range of software types, nor software license types, created and used by the FOSS Community.

may seem revolutionary from the perspective of those of us most familiar with the proprietary software world, this concept of open sharing of information – and indeed of source code – is not new. One commentator has noted that:

> Science is ultimately an Open Source enterprise. The scientific method rests on a process of discovery, and a process of justification. For scientific results to be justified, they must be replicable. Replication is not possible unless the source is shared: the hypothesis, the test conditions, and the results. . . .Ultimately, the Open Source movement is an extension of the scientific method, because at the heart of the computer industry lies computer science.[2]

FOSS started in the early 1980's as a movement among individual computer scientists and software engineers, concentrated at universities or other "think tanks", who proudly called (and call) themselves "hackers". Now, far from being limited to a movement consisting of individual developers, the FOSS movement has grown and includes a wide variety of players, including commercial and non-commercial enterprises, FOSS distributors and service providers and not-for-profit open source projects that foster the development of a particular type of software or FOSS in general. In the early days of the FOSS movement, FOSS was often dismissed as being just a "hacker thing" or "a phase" that wouldn't go anywhere because "you can't make money from it". But now, any technology entity that seeks to avoid FOSS or minimize its infiltration into the commercial enterprise does so at its peril.

Why Should Companies Pay Attention? Because FOSS is Everywhere.

While early FOSS development was more concentrated at universities and academic labs, more and more of those computer science students and academic hackers now have left school to work in industry. These programmers not only may be bringing code into their employer companies without management's necessarily knowing about it, but they also may be releasing code via contributions to open source projects without any company authorization. As Chris DiBona pointed out some years ago:

> We have reached the stage where an entire generation of students who learned computer science under the influence of [FOSS] is now at work in industry, and have quietly been bringing free software in through the back doors of industry for years. They do so not from altruistic motives, but rather to bring better code to their work. [3]

This ability and desire of software engineers to bring software into their companies freely and without scrutiny raises particular concerns when that software arrives in source code, rather than object code, form. When software comes in as source code, developers' ability to modify and create derivative works from that software increases. This means that software is easier to commingle with proprietary code – including a company's "home-grown" code and third party code. The consequences of such commingling – particularly with respect to FOSS code governed by the "copyleft" license provisions discussed here – can be severe. Possible consequences include the unintended loss of

---

[2] DiBona, "*Introduction*" in DiBona, Ockman & Stone (editors); Open Sources: Voices from the Open Source Revolution, (O'Reilly 1999) pp. 1-17, at 7.

[3] See DiBona in DiBona*, et al,* at p. 5.

critical company intellectual property and infringement claims from FOSS or other third party software providers, if the commingling has triggered requirements to make company and other proprietary source code publicly available upon distribution of commingled code.[4] One of the "worst nightmares" surrounding FOSS is the story of Linksys, which Cisco purchased in 2003 for U.S. $500 million. Three months after the purchase, complaints appeared on FOSS discussion boards claiming that Linksys had violated a key FOSS license by not releasing source code found in its product. Several months and a lot of negotiation later, Linksys publicly released the subject source code.[5] One lesson from the Linksys story is that companies need to be alert to the possibility that they have FOSS in their code bases already, like it or not. And that they may already have unwittingly donated some of their intellectual property to the broader community. But as discussed below, rather than panicking, companies can take steps to manage FOSS within their organizations so as to reap benefits from it without exposing themselves to undue risk.

Why Should Companies Pay Attention? Because FOSS Might Be Profitable.

The risks of dealing with FOSS can seem daunting, and they are the reason that so much "Fear, Uncertainty and Doubt" ("FUD") surrounds its use and distribution. However, there are more positive reasons that companies in the proprietary space are – and should be -- paying attention to FOSS. For one, FOSS now is an important force in the economy. More and more companies are realizing that a route to developing better and more innovative technology, for less money and in less time, is by leveraging FOSS in their product lines and company infrastructures. Many companies are recognizing that FOSS may offer them a way to promote standards that facilitate broader adoption of company proprietary technology. And companies also increasingly are realizing that supporting the FOSS Community can lend significant marketing advantages through increased name recognition and can project the image of the company as a "good corporate citizen". In other words, engaging with FOSS and the FOSS Community can be – and increasingly is -- a way for commercial entities to open up a broad new range of business and technical opportunities, thereby increasing profits.

Indeed, many individuals and companies now are structuring entire businesses exclusively around FOSS – and are making money -- in ventures that may include compiling complex code into neat distribution packages that make FOSS more accessible, or providing consulting and technical support services focused on FOSS. [6] Whole new industries have developed around FOSS, such as the fledgling but rapidly growing business around source code audit tools and tracking services that help

---

[4] See Murray "*Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise*" (October 2005) pp. 5-6 available at http://www.mataulegal.com/articles.htm

[5] For a discussion of this event, see Meeker, "The Legend of Linksys" in LinuxInsider (January 1, 2006), available at http://www.linuxinsider.com/story/47987.html

[6] Examples of commercial companies whose businesses are based upon FOSS are Red Hat, Inc., which offers Linux distributions for the enterprise and related services (http://www.redhat.com) and the Olliance Group, which offers consulting services relating to FOSS (http://www.olliancegroup.com/).

companies determine whether they have "viral" or other potentially problematic code in their proprietary code bases.[7]

Why Should Companies Pay Attention? Because Everybody's Getting with the Program.

As noted earlier, the advantages and infiltration of FOSS increasingly are recognized, and supported, by major commercial ventures. Some of the best known and popular FOSS projects and initiatives are openly funded in the millions of dollars by large companies whose businesses traditionally have been "proprietary". These companies not only seek to fund software development and innovation to further their technical needs and improve the products they market, but they also can derive tremendous marketing advantages from their prominent support of FOSS.[8] 2005 in particular was a year for the growth of FOSS code in mainstream business, with companies increasingly using high-quality, freely-available code to run their IT infrastructures, as well as to build commercial applications that integrate their own proprietary intellectual property with open source software. Revenues derived from Linux, the best-known FOSS application, are at approximately $25 billion at time of writing, and are expected to reach $35 billion by 2008.[9]

The amount of FOSS now available, and the corresponding number of FOSS licenses out there, has mushroomed – and those numbers are growing rapidly and constantly. Now, FOSS and other freely available software can be downloaded from – and posted to -- open source software repositories such as SourceForge.net, commercial proprietary software companies (such as Microsoft, Sun and Oracle), open source projects (such as Apache, JBoss and PostgreSQL), university and personal websites, and commercial open source companies (such as Sleepycat and MySQL), among other sources. When checked recently, the number of FOSS projects hosted on Sourceforge

---

[7] See, for example, the products and services offered by Black Duck Software, Inc. at www.blackducksoftware.com and Palamida, Inc at www.palamida.com .

[8] As examples, as of this writing, Eclipse.org, an open source community which " whose projects are focused on providing an extensible development platform and application frameworks for building software" lists "Strategic Members" that include Borland, CA, IBM, HP, Intel, Sybase, and SAP (see http://www.eclipse.org/membership/members/strategic.php). The Open Source Development Labs ("OSDL"), a non-profit organization whose stated mission is to "accelerate the deployment of Linux for enterprise computing" lists as members such companies as CA, HP, Hitachi, IBM, Intel and NEC (see http://groups.osdl.org/osdl_members/osdl_roster)

[9] Statistics from IDC and VDC kindly provided to the author by William Weinberg , Analyst/Strategist at OSDL, also indicating that Linux market share for desktop applications was approaching 5%, for data center applications hovered around 10%, and for embedded applications were approaching 30% in 2005. For more information, see OSDL/IDC study results summarized at http://www.osdl.org/docs/linux_market_overview.pdf . See also http://news.netcraft.com/archives/web_server_survey.html . In reporting responses from approximately 75 million websites, Netcraft.com reported that approximately 52 million of those (or 70%) were running using Apache web server in January 2006.

alone was more than 109,000, with the number of registered users listed as well over 1.2 million.[10] As was pointed out in 2003,

> One of the most widely used Web server programs in the world (Apache), one of the dominant Web programming languages (Perl), the program that routes more than 80 percent of all Internet email messages worldwide (Sendmail), the program that is the basis for the domain name system (BIND), and the fastest growing operating system in the world (Linux) are all open source.[11]

And, as Chris DiBona noted back in 1999:

> . . .the Internet is built on a powerful collection of open standards maintained on the merit of individual participation, not the power of a corporate wallet. The Internet is, in many ways, the original Open Source venture. Keeping the Internet firmly based on open standards made it possible for a wide and diverse range of programmers to work on developing Internet applications. The Internet's spectacular growth is a testament to the power of this open standards model.[12]

In short, as long as there is an Internet, FOSS is here to stay.

THE FOSS COMMUNITY

As noted above, for software to be considered "free" or "open source" within the FOSS community, its source code must be freely available – and its redistribution free of charge. But not all "open source" software meets this definition. Software may be distributed in source code form, but licensed with restrictions preventing its redistribution or its commercial use. For purposes of this paper, "FOSS" does not include such proprietary code issued with restrictions on redistribution or commercial use, nor does it include software released into the public domain. While that kind of licensing may provide for "openly" licensed source code, it does not meet the definitions created by the FOSS Community.[13]

---

[10] See http://sourceforge.net/. Last checked on January 8, 2006, when it listed 109,723 Registered Projects (up from 103,804 exactly three months earlier) and 1,216,034 registered users (up from 1,152,363 three months earlier).

[11] Wacha,"Open Source, Free Software, and the General Public License", *10 The Computer Lawyer 3 (March 2003)*.

[12] DiBona, in DiBona, *et al*, at p. 15

[13] See, for example, comments by Richard Stallman that

> If a program is free software when it leaves the hands of its author, this does not necessarily mean it will be free software for everyone who has a copy of it. For example, public domain software (software that has not been copyrighted) is free software; but anyone can make a proprietary modified version of it. Likewise, many free programs are copyrighted but distributed under simple permissive licenses that allow proprietary modified versions.

Stallman, "*The GNU Operating System and the Free Software Movement"* in DiBona *et al*, pp. 53-70, at pp. 58-59.

Keeping the perspective of the FOSS Community in mind is critical when entering the realm of FOSS. The existence and influence of "the Community" is perhaps the most important distinction between the world of FOSS world and that of proprietary software. Although FOSS licenses are prolific and pervasive, their enforceability is only now beginning to be tested in the courts. There is not settled "law" around open source software, but there is increasingly settled practice and enforcement given the high rate of growth of Linux and other popular open source programs, and the outspoken and passionate nature of the FOSS Community. Anyone, or any entity, that proposes to venture into the world of FOSS, needs to understand that

> . . . it is the 'elaborate set of customs' in the community that provide the strongest guidance (whether politely or otherwise). A software provider that the community perceives is doing 'right' may read its praises on the Web. A provider perceived as doing wrong, however, will have negative 'flames' spread worldwide.[14]

FREE VS. OPEN SOURCE SOFTWARE

To understand what FOSS and the FOSS Community are all about, it may be helpful to understand some of their history. Prior to the times when computers and software began to be mass-marketed to the public, most software development occurred in an "open source" environment. Programmers working on large computer systems generally worked together on creating code that would run those systems, in a collaborative manner. However, as computers were used more prevalently by the general public, and particularly with the rise of the personal computer and mass-marketing of computer products, the software used to run those computers increasingly was developed as proprietary intellectual property. This proprietary and trade secret code was, and continues to be, licensed under restrictive license terms that limited its use, copying, modification and distribution – and that usually conditioned any or all of these limited rights on the payment of money.

Richard Stallman, a researcher at the MIT AI Lab and now well known as the founder of the GNU project[15] and of its umbrella organization, the Free Software Foundation ("FSF"), saw this guarding of source code as trade secret property as a threat to innovation. In what he has described as a "stark moral choice",[16] Stallman spearheaded the Free Software movement in an effort to keep source code available to the general public. He believed that "source code is fundamental to the furthering of computer science and freely available source code is truly necessary for innovation to continue".[17] Other hackers have joined Stallman in what has become known as "the

---

[14] Wacha, "Open Source, Free Software, and the General Public License", *10 The Computer Lawyer 3 (March 2003)*.
[15] GNU stands for "GNU's Not Unix".
[16] Stallman, "*The GNU Operating System and the Free Software Movement*"; in DiBona *et al* pp. 53 – 70, at p. 55.
[17] DiBona in DiBona et.al, at p. 2

Community". This "Community" has assumed the role of chief license author, license interpreter, and license enforcer when it comes to the licensing and distribution of FOSS. Other players, including lawyers and the courts, have been noticeably absent from the scene until relatively recently.

"Open Source" Software, though similar to "Free Software" for legal purposes, evolved as a result of the 1997 split-off from the Free Software movement by members of the FOSS Community who feared that the "viral" characteristics of Free Software would restrict FOSS from being used by commercial enterprises – and, consequently, would limit the growth and adoption of FOSS. This group of leaders in the free software community "were concerned that the Free Software Foundation's anti-business message was keeping the world at large from really appreciating the power of free software."[18] They embarked upon a marketing campaign to promote free software, which led to the development and use of a new term to describe the software they were talking about: "Open Source". The "Free" and "Open Source" software movements share common goals, in that they both focus on the free redistribution of programs and the requirement to make source code available. However, these movements grew from different perspectives, and the terms "Free" and "Open Source" software do not necessarily mean the same thing. This distinction is explained in more detail below.

"Free" Software

The term "Free" software means that the licensee has the freedom to run, modify and distribute the software. However, "Free" does not necessarily mean "without cost", as Stallman has explained:

> Since "free" refers to freedom, not to price, there is no contradiction between selling copies and free software. In fact, the freedom to sell copies is crucial: collections of free software sold on CD-ROMs are important for the community, and selling them is an important way to raise funds for free software development. Therefore, a program that people are not free to include on these collections is not free software.[19]

At the FSF's website where its philosophy and definition of free software is provided, the FSF clarifies that one should think "free speech, not free beer".[20] The FSF further explains that "free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software." It specifies "four kinds of freedom" for the users of Free Software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).

---

[18] DiBona, *et al*. p. 3

[19] Stallman, "*The GNU Operating System and the Free Software Movement*", in DiBona *et al* pp. 53 – 70, at p. 56.

[20] See www.gnu.org/philosophy/free-sw.html

- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this."[21]

The Free Software movement generally is considered either a greater promoter of freedom, or more restrictive of licensees' freedom, than the Open Source movement -- depending upon one's perspective. This is not just because of the strongly-held views of its founder. Under FSF guidelines as to what constitutes "free software", *all modifications to the licensed software must be licensed under the same terms under which it originally received*. This kind of requirement is known as "copyleft", "viral" or "reciprocal". And this kind of requirement is what understandably strikes most fear into the hearts of the commercial enterprise. If "viral" FOSS comes into a company without proper attention to the licensing terms that govern it, a company's entire code base – or that of a key product line – could be "contaminated" and required to be released to the general public. And this risk is exacerbated by the fact that, given the vigilance and outspoken nature of the FOSS Community, that requirement might be raised and enforced in a very public way. As Heather Meeker has noted ". . .in the free software world, while people may rat on you because it is in their personal interest, they will also rat on you because it is their moral duty." And this kind of enforcement by "true believers" can be more "perilous" than that initiated by competitors or others. [22]

"Open Source" Software

"Open Source" Software is similar to "Free Software" in that its licensees are entitled to access, use, copy, modify and distribute OSS source and binary code without making royalty payments to the licensor, and to combine OSS with other software code. But software that is defined as "Open Source" may be licensed with fewer restrictions (or freedoms, depending upon one's perspective!) on its downstream distribution. Open Source" licenses may be "copyleft" licenses but, alternatively, may permit the licensee to take his/her modifications to the code private (we refer to this kind of license as an "Academic" or "Non-viral" license). For this reason, and given its founders' desire to support commercial adoption of FOSS, "Open Source" software generally is seen as more commercial-leaning than is "Free Software".

There are various definitions of "Open Source", but the one most referred to (and originally composed by Bruce Perens, one of the founders of the "Open Source" movement), is the Open Source Definition ("OSD") created by the Open Source Initiative ("OSI"). The OSD clarifies that "[o]pen source doesn't just mean access to the source code." Under the OSD, the distribution terms of open-source software must comply with the following ten criteria in order to qualify as "Open Source":

[21] See www.gnu.org/philosophy/free-sw.html
[22] Meeker, "The Legend of Linksys" in LinuxInsider (January 1, 2006), available at http://www.linuxinsider.com/story/47987.html

1. Free Redistribution

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

2. Source Code

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

3. Derived Works

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

4. Integrity of The Author's Source Code

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

5. No Discrimination Against Persons or Groups

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

6. No Discrimination Against Fields of Endeavor

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

7. Distribution of License

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

8. License Must Not Be Specific to a Product

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

9. License Must Not Restrict Other Software

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

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.[23]

The OSI determines whether licenses comply with the OSD. If it determines that they do, they are designated (and marketed) as "OSI Certified™ open source software. The Open Source Initiative currently lists close to seventy such "OSI certified" licenses.[24] Though OSI-certification lends certain legitimacy to FOSS licenses within the FOSS Community, the OSD is not the only definition of what constitutes "open source" software. For example, Larry Rosen presents five principles of "open source" in his recent book "Open Source Licensing: Software Freedom and Intellectual Property Law":

- Licensees are free to use open source software for any purpose whatsoever
- Licensees are free to make copies of open source software and to distribute them without payment of royalties to a licensor
- Licensees are free to create derivative works of open source software and to distribute them without payment of royalties to a licensor
- Licensees are free to access and use the source code of open source software.
- Licensees are free to combine open source and other software.[25]

In other words, licenses that are not OSI-certified still may be bona fide "open source" or free licenses.

What is Not FOSS.

It may be important to note that not all "open source" software is released under OSD or FSF-approved terms. While many software projects release their software under relatively well-known and well-understood licenses such as the GNU General Public License (the "GPL") or the Berkeley Software Distribution ("BSD") license, a significant number of software providers release software under different license terms that may not meet accepted definitions of "open source".[26] Some software is downloadable without a

[23] See www.opensource.org/docts/definition_plain.html

[24] See http://www.opensource.org/licenses/ Last checked on January 15, 2006.

[25] Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, (Prentice Hall PTR 2005), pp. 9-11.

[26] Various definitions of what "open source" means exist. One of the most commonly referred to is that of the Open Source Initiative, which can be found at http://www.opensource.org/docs/definition.php. Readers may also be interested in the discussion of the difference between "open source" and "free software" found at http://www.gnu.org/philosophy/free-software-for-freedom.html .

clear path to the applicable license terms, or possibly with no license terms at all.[27] Many software developers have chosen to issue their software under terms they have crafted themselves or created by modifying existing licenses[28] – and which terms or licenses do not appear on the OSI list nor necessarily fit accepted definitions of FOSS.

FOSS vs. "CLOSED SOURCE" OR "PROPRIETARY" SOFTWARE

Before diving further into particulars of FOSS licensing, it may be helpful to consider how FOSS both is similar to, and different from, "closed source" or "proprietary" software. In most respects FOSS and proprietary software licensing are more similar than different.

For one thing, the development processes for each of these kinds of software consist of common elements. As Paul Vixie has pointed out, the engineering process, whether for proprietary or open source software, is essentially the same: Developers need to

- Identify a requirement, and its requirers
- Design a solution that meets the requirement
- Modularize the design and plan the implementation
- Build it, test; deliver it; support it[29]

And, to make money from either kind of software, the same principles of good business strategy apply. Robert Young, a founder of Red Hat Software, has said that:

> While making money with free software is a challenge, the challenge is not necessarily greater than with proprietary software. In fact you make money in free software exactly the same way you do it in proprietary software: by building a great product, marketing it with skill and imagination, looking after your customers, and thereby building a brand that stands for quality and customer service.[30]

---

[27] For example, see http://www.qmail.org/top.html, which states that "you can. . .redistribute qmail for free" but includes very specific limitations on any modification, including getting Dan Bernstein's prior approval "of the exact package that you want to distribute." See http://cr.yp.to/qmail/dist.html. It has been argued that these terms do not constitute a license at all. See Rick Moen's discussion of Dan Bernstein's "license" at http://linuxmafia.com/~rick/faq/index.php?page=warez#djb and Mr. Bernstein's response to Mr. Moen's comments at http://cr.yp.to/distributors.html .

[28] See, for example, the gSOAP public license, which includes this helpful introduction: "The gSOAP public license is derived from the Mozilla Public License (MPL1.1). The sections that were deleted from the original MPL1.1 text are 1.0.1, 2.1.(c),(d), 2.2.(c),(d), 8.2.(b), 10, and 11. Section 3.8 was added. The modified sections are 2.1.(b), 2.2.(b), 3.2 (simplified), 3.5 (deleted the last sentence), and 3.6 (simplified)." See http://www.cs.fsu.edu/~engelen/license.html

[29] Vixie, "*Software Engineering*", in DiBona *et al*, pp. 91-100 at p. 99

[30] Young; "*Giving It Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry*" in DiBona, et. al., pp. 113-125, at p. 114

Finally, both licensing schemes are based upon principles of copyright law, which give the author of a work (the copyright holder) the ability to determine the terms on which his or her work may (or may not) be copied, modified and distributed.

Proprietary Software

A major difference between FOSS and "closed source" programs is how the software itself is developed. In the proprietary world, software is developed in secret, by developers working for commercial companies that make money by keeping their software code and other technology closed, so that they can charge money (royalties) for its use and distribution. Accordingly, their developers are bound by employee nondisclosure agreements not to share the software code they create or see with anyone outside the company. And those same developers are assumed to be motivated by the promise of getting rich through stock options or other benefits more than by anything else.

Given the closed nature of the proprietary development process, while there may be some peer review of software that is developed by and within the company's own engineering team, there generally is no independent peer review of the software developed within the company. Field testing is carried out through internal testing and some external beta testing, but software generally is not released for commercial use by the intended users until these tests are completed. Support is carried out by internal technical support personnel, or by independent contractors, who are responsible for fixing reported bugs and maintaining the software – all in exchange for payment of money.

Further, the licenses that permit the use, copying and/or modification of proprietary software generally:

- Make source code unavailable (except under a specific (and likely restrictive) source code license agreement or via a source code escrow arrangement);
- Don't let licensees acquire, install, use, copy or modify software without paying money; and
- Impose downstream restrictions such as prohibitions on sublicensing, copying, redistributing, or modifying the software

Another characteristic of proprietary software licensing is that the applicable license terms usually are kept confidential. Unless a company's licensee or acquirer requires that it make representations and warranties regarding incoming code, or otherwise for marketing and publicity reasons, the terms on which its software code has been obtained are kept secret. Terms may become more public during enforcement actions relating to permitted software use and other license terms but, in most cases, litigation is pursued by individuals or companies through the courts without much public scrutiny. Since license transactions are primarily motivated by revenue, disputes often are resolved through payment of money. And, unless there is a hotly-contested lawsuit

between major players that is resolved via litigation of public interest, infringement claims and settlement outcomes are not publicized. [31]

FOSS Programs

In contrast, FOSS programs often are created (and modified) in a collaborative development environment, where individual developers post their contributions of software code as they are created to publicly accessible source code repositories, such as Sourceforge, specific software projects such as Apache or PostgreSQL, or academic sites from which and to which software can be downloaded. This software can be accessed and modified by anyone with internet access, and users who find and fix bugs or make other improvements are able to, and encouraged to, contribute their improvements and modifications back to the software project. This open process of "peer review" throughout the life cycle of FOSS software has led some commentators to the conclusion that "open-source software enjoys the best system-level testing in the industry".[32]

Another distinction between FOSS and proprietary software development is that motivations of the developers creating and improving FOSS may be very different than those contemplated by the proprietary software world. Rather than writing great software for financial reward, FOSS developers generally do so because they care about and want to enhance their reputations as great programmers – and, perhaps more importantly, because they have a passion for programming and for the particular software. As Chris DiBona put it:

> Much like the rush a runner feels when running a race, a true programmer will feel this same rush after writing a perfect routine or a tight piece of code. . . The point is that many programmers code because it is what they love to do, and in fact it is how they define their intellect. . . [O]ther programmers consider themselves, rightly, to be scientists. Scientists aren't supposed to hoard profits from their inventions, they are supposed to publish and share their inventions for all to benefit from.[33]

And the license terms governing FOSS, while containing many of the same categories of provisions as those for proprietary software regarding permitted use and distribution, attribution, warranty (or lack thereof), are different from proprietary licenses in that they generally:

- Prohibit payment of money for use, copying, modification or redistribution, or else permit the licensee to do those things without paying money; and
- Permit or require redistribution under terms that make the source code available

[31] For further discussion of this point, see Murray, "*Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise*", (October 2005) at p. 9 available at http://www.mataulegal.com/articles.htm
[32] Vixie, "*Software Engineering*", in DiBona *et al*; p. 98
[33] DiBona at p.13

Finally, in the world of FOSS, not only are license terms out there on the web for everyone to see, but the entire FOSS Community takes it upon itself to interpret and enforce those terms. Members of the FOSS Community are committed to the promotion of open source software, communicate amongst themselves frequently, and see as part of their mission promoting the cause of FOSS and helping enforce license terms. One of the most common introductions to message or blogs from FOSS Community members about license terms is "IANAL but. . ." – meaning "I am not a lawyer, but. . ." Long threads on http://slashdot.org/ are devoted to happenings of interest to the FOSS Community, including notifying others of potential license violations. So, while individual licensors and licensees may raise issues via the courts, many FOSS-related disputes to date (like that of Linksys) have been resolved informally with the "help" or "interference" -- depending upon one's perspective -- of active members of the FOSS Community, including the FSF.

Thus, far, the FSF has enforced the GPL informally via threats of exposing license violators to bad publicity. As the FSF's General Counsel, Eben Moglen, has explained:

> . . .no one wanted to be seen as the villain who stole free software, and no one wanted to be the customer, business partner, or even employee of such a bad actor. Faced with a choice between compliance without publicity or a campaign of bad publicity and a litigation battle they could not win, violators chose not to play it the hard way.[34]

Elaborating on why the FSF's strategy of pressuring companies into compliance has worked and why it has not seen the need to pursue litigation in the courts to enforce the GPL, Moglen says that any company that doesn't respond will find itself in this situation:

> . . .that [company's] customers are going to go elsewhere, talented technologists who don't want their own reputations associated with such an enterprise will quit, and bad publicity will smother them. And that's all before we even walk into court.[35]

Bad publicity, via postings by FOSS Community members on public websites and chatrooms such as Slashdot, is one of the most-feared outcomes of companies working with FOSS – and another aspect of FOSS that generally does not play into the world of proprietary software licensing. It may be hard to prove innocence when guilt is presumed, and guilt may be presumed in a very public – and sometimes a very nasty – way. Among other outcomes, this kind of publicity can create a public relations nightmare for commercial companies.[36]

---

[34] Moglen, "*Enforcing the GNU GPL*"; (10 September 2001), located at http://www.gnu.org/philosophy/enforcing-gfpl.html

[35] Moglen; "*Enforcing the GNU GPL*"; (10 September 2001), located at http://www.gnu.org/philosophy/enforcing-gfpl.html

[36] For further discussion of this point, see Murray, "*Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise*"(October 2005) at p. 9, available at http://www.mataulegal.com/articles.htm

TYPES OF FOSS LICENSES

As indicated earlier, there are two general categories of FOSS licenses: what we call here "Nonviral" or "Academic" licenses, and those we refer to as "Copyleft", "Viral" or "Reciprocal" licenses. This paper does not go into detail about the various licenses, but offers a brief discussion of their general nature and of a few of the more popular licenses.

"Non-viral" or "Academic" Licenses

Examples of FOSS licenses that are "non-viral" or "academic" are the MIT, BSD (an abbreviation for "Berkeley Software Distribution"), Apache and the Academic Free ("AFL") licenses. With the exception of the latest version of the Apache License (v 2.0) and the AFL, these are some of the earliest open source licenses, and also are the simplest and easiest to understand. The essence of the BSD license is found in one line: *"Redistribution and use in source and binary forms, without or without modification, are permitted. . . "*[37] In other words, the licensee essentially can do whatever it wants with the licensed code as long as attribution and warranty disclaimer notices are provided with any distribution. The Apache v.2.0 and the AFL licenses, more recently developed, are more complex. They include most of the same provisions of the MIT, BSD and earlier Apache licenses, but they also include other clauses such as express patent licenses and "patent defense" clauses that terminate the license if the licensee files a patent infringement claim regarding the licensed work.[38] These kinds of clauses are discussed in more detail below.

These "academic" licenses, as applied to the originally licensed code, allow that software code to be used in any way that the licensee chooses: in other words, these licenses permit the licensee to use, create derived works from and distribute the original or derived code under proprietary, or "closed" licenses as well as under FOSS licenses. Accordingly, these licenses are used frequently by commercial enterprises, often with the result that the code is "forked" and derivative works are lost to the open source community. Given their flexibility as to downstream use and distribution, however, the "academic" licenses also are compatible with most open source licenses and are widely used by the FOSS Community.[39]

"Copyleft" or "Reciprocal" Licenses

"Copyleft", "Viral" or "Reciprocal" licenses differ significantly from the "Academic" or "Non-viral" licenses described above, in that they place specific limitations on the relicensing or distribution of the subject code and any derivative works.

---

[37] See http://www.opensource.org/licenses/bsd-license.php

[38] See Apache v. 2.0, Section 3 at http://www.opensource.org/licenses/bsd-license.php and Academic Free License v. 2.1, Sections 2 and 10 at http://www.opensource.org/licenses/afl-2.1.php .

[39] Although note that the FSF maintains a list of licenses that it considers not compatible with the GPL that includes the original BSD license, as well as the Apache and AFL licenses. See http://www.fsf.org/licensing/licenses/index_html#GPLIncompatibleLicenses

These "Copyleft" licenses include the GPL, the Lesser GNU General Public License ("LGPL"), the Mozilla Public License ("MPL") and the "Common Public License ("CPL"). All of these licenses are OSI-Certified as meeting the OSD, and all are considered "Free Software" licenses by the FSF – although the last three of these are listed as "GPL-Incompatible Free Software Licenses" on the FSF's website.[40]

While each of these licenses differ considerably in terms of the scope of their "viral" nature, their general nature and intent is exemplified by the central GPL Provision that states:

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.[41]

The GNU General Public License (the "GPL")

The GPL, like the BSD, is one of the original and most frequently used open source licenses. While the license permits anyone to use and modify code licensed under the GPL essentially without restriction, it *requires any code originally licensed under the GPL or derived therefrom to be redistributed under the GPL as well*. In short, while the GPL permits anyone to use or modify GPL'd code for his/her own use in any way he or she desires, the GPL "does not permit the creation of derivative licenses from the license itself."[42] Richard Stallman has explained the reasoning behind this aspect of the GPL:

> Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free. The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program and distribute modified versions – but not permission to add restrictions of their own. Thus, the crucial freedoms that define "free software". . . become inalienable rights.[43]

Notwithstanding the terms of the GPL that require redistribution of GPL'd and derived software "at no charge" or "free", distributors are permitted to charge for any services they provide in connection with a redistribution or sublicensing – including any fees they wish to charge for compilation of code onto media or distribution of media. Also, the requirement to relicense under the same GPL terms does not mean that any redistributor of code or derived code must release source code to the entire world. Rather, if a sublicensee receives a GPL'd program in binary form, that sublicensee has the right to receive the source code for a period of up to three years. So, rather than providing source code, the distributor or provider simply may provide the right and

---

[40] See http://opensource.org/licenses/ and http://www.fsf.org/licensing/licenses/index_html#GPLIncompatibleLicenses

[41] See http://www.gnu.org/licenses/gpl.html , Section 2

[42] St. Laurent, Understanding Open Source & Free Software Licensing; (O'Reilly 2004) p. 35

[43] Stallman,"*The GNU Operating System and the Free Software Movement*"; in DiBona *et al* pp. 53 – 70, at p. 59.

means to obtain the source code during that three-year period at no charge other than the cost of delivery.[44]

Given FSF philosophy, and the views of its founder, the terms of the GPL are not expected to change substantially with time. However, readers should be aware that, as of January 16, 2006, a new version 3.0 of the GPL is in draft and has been released for public.[45] That draft version specifically addresses issues such as international enforceability of the GPL and patent rights, among other issues. Based on informal discussions the authors have had with members of the FOSS Community, the comment and review process of this new GPL version is expected to last at least a year before the document is finalized.

The Mozilla Public License (the "MPL")

As the GPL evolves with the growth of FOSS in the commercial context, so have a variety of other licenses. One of the most important of these is the Mozilla Public License, or MPL, which was developed by Netscape (along with its predecessor license, the Netscape Public License) to foster ongoing development of its proprietary browser technology. Netscape chose to write a new license rather than issue its code under the BSD, GPL or other existing licenses for reasons that are explained on the www.mozilla.org website. In Netscape's view, the BSD license did not go far enough to ensure that developers would contribute their changes to the Netscape Communicator back to the community. Netscape believed that such re-contribution was important for the long-term viability of the project. Additionally, Netscape chose not to use the GPL to govern its open source project and licensing because use of the GPL would conflict with obligations Netscape had relating to third party components of the Communicator product, and also potentially with U.S. legal requirements relating to cryptographic code -- and therefore could restrict the ability to distribute the code worldwide. Further, Netscape wanted to be sure that it could reserve the rights not to distribute source code for other Netscape products in which it might incorporate portions of the Communicator code. Finally, Netscape believed that use of the GPL might discourage the use and development of its code by other commercial developers, so it sought to create a "less restrictive license".[46]

The MPL's "viral" or "copyleft" provisions are not as extreme as those of the GPL or LGPL, and are based around the definition of a "Modification" and the notion of licensees creating and maintaining separate "files" containing code that may or may not be subject to the MPL. As Mozilla explains, a "Modification" is any change to MPLed files, or new files into which MPLed code has been copied. These Modifications must be released under the same MPL License. However, new files that contain only the licensee's code are not Modifications, are not covered by the MPL, and may be kept

[44] See Wacha; "Open Source, Free Software, and the General Public License"; *10 The Computer Lawyer 3 (March 2003)*
[45] See http://gplv3.fsf.org/
[46] See http://www.mozilla.org/MPL/FAQ.html

proprietary.[47] In other words, even if a licensee has created a derivative work by adding its own files to the work, any files not containing MPLed code or Modifications to that code are not affected by the reciprocity, or copyleft, provisions of the license. As Larry Rosen has explained:

> Reciprocity under the MPL is defined narrowly so as to encourage the use of open source software as building blocks to create Larger Works. . . .Those Larger Works may be open or proprietary; with respect to them, the MPL acts like an academic license. But the individual building blocks are licensed with reciprocity obligations. If you distribute improvements to those building blocks, you must license those improvements under the MPL as open source software.[48]

The MPL includes a number of provisions not found in previous licenses such as those relating to indemnities, government rights, jurisdiction, and attorneys' fees. But perhaps most notable is that the MPL was one of the first licenses to include substantive clauses addressing specific intellectual property rights, including patent licenses and a "patent defense" provision. The BSD license is silent on the issue of patents, and the current version of the GPL (v. 2.0) does not deal with patents other than stating that, if a court judgment or patent issues prevents compliance with the license, then the licensed program may not be distributed.[49] But the MPL includes specific licenses of patent claims to licensees with respect to the unmodified originally licensed code when distributed on a standalone basis, without limiting the original licensor's right to exclude others from making, using or selling other software. The MPL also includes a reciprocal license from each "Contributor" with respect to patent claims.[50] Given the various exclusions from the MPL patent license grants, however, potential licensees should carefully review the MPL license terms and consider whether they may need to seek additional patent licenses prior to working with or distributing MPLed code.

Under the MPL, licensees are required to post notices relating to third party patent claims of which they are aware and that relate to the licensed code. The MPL also includes provisions that terminate the license grants if a licensee brings certain kinds of patent litigation. These "patent defense" provisions apply only to patent claims, and not to claims alleging infringement of other types of intellectual property such as copyright and trademark. With respect to patent claims relating to the licensed MPLed code, this termination of rights is prospective only, creating no liability for use of code prior to termination, and includes a 60-day "cooling off period" during which the claim can be withdrawn or otherwise resolved. Interestingly, though, the MPL also includes a more draconian termination provision with respect to claims for patent infringement that do not relate to code covered by the license. If a licensee sues the Initial Developer or a Contributor for patent infringement with respect to technology "other than such

---

[47] See http://www.mozilla.org/MPL/mpl-faq.html

[48] Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, (Prentice Hall PTR 2005) p. 147.

[49] The GNU General Public License v 2.0, Section 7, available at http://opensource.org/licenses/gpl-license.php

[50] The Mozilla Public License v 1.1, Sections 2.1 and 2.2, available at http://www.mozilla.org/MPL/MPL-1.1.html. For a discussion of these provisions, also see Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, (Prentice Hall PTR 2005) pp. 148 - 154.

Participant's Contributor Version", any patent rights granted under the license to that licensee are revoked retroactively and without any opportunity for cure.[51] Again, prior to working with MPLed code, potential licensees should review their patent portfolios to consider whether these patent defense provisions are likely to be problematic.

The Common Public License ("CPL")

Another of the principal commercial licenses is the Common Public License, or CPL, developed by IBM. IBM has been an active participant in the open source community as a participant in key open source projects such as Linux and Apache, and has been a prominent supporter of organizations such as the Open Source Development Labs ("OSDL"). In early 2005, IBM received lots of attention by making 500 of its software patents available for use by developers of open source software that is subject to OSI-certified licenses. This "patent grant" has been followed, in varying forms, by a number of other companies. [52]

The CPL is a variant of IBM's Public License, one of the vendor-specific licenses approved by the OSI. It also has served as a basis for at least one other license, the Eclipse Public License which, with minor modifications, is almost identical to the CPL.[53] The CPL removes most of the IBM-specific terms from the IBM Public License, resulting in a template license that other companies and organizations can use to govern distribution of open source software. "Contributors" under the CPL grant to "Recipients" all rights to their "Contributions" necessary for open source software, including an express patent license with respect to the Contribution. However, as with the MPL, the patent license of the CPL excludes combinations of the licensed software with other software or hardware. The CPL license also does not apply to patents not issued at the time a Contribution is added – in other words, the license is limited to those uses and combinations contemplated at the time the Contribution first is made.[54] As with the MPL, therefore, potential licensees need to consider whether the patent licenses granted under the CPL serve their needs – and whether the patent defense clauses summarized below raise particular concerns given their patent portfolios.

The CPL's reciprocity obligations also differ from those in the GPL and MPL, although in effect they have similarities to both licenses. The CPL permits a licensee to distribute its derivative works ("Programs") in object code form under its own license

[51] The Mozilla Public License v 1.1, Section 8.2, available at http://www.mozilla.org/MPL/MPL-1.1.html. For a discussion of these provisions, also see St. Laurent, Understanding Open Source & Free Software Licensing, (O'Reilly 2004) p. 77

[52] See, for example, "*IBM Pledges Patents to Open Source*" from January 11, 2005 at http://www.internetnews.com/dev-news/article.php/3457381 and the discussion of IBM and other companies' patent grants and of software patents generally by Bruce Perens in "*Perspective: The Open-Source Patent Conundrum*" from January 31, 2005 at http://news.com.com/The+open-source+patent+conundrum/2010-1071_3-5557340.html

[53] See Eclipse Public License v 1.0 at http://opensource.org/licenses/eclipse-1.0.php

[54] See Common Public License v. 1.0, Section 2, available at http://opensource.org/licenses/cpl1.0.php
Also see discussion in Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, (Prentice Hall PTR 2005) pp. 163-167.

agreement, but only if that license agreement provides for availability of attendant source code. Derivative Works distributed in source code form must be distributed under the terms of the CPL. However, the CPL offers an important exclusion from the "copyleft" provisions of the license: ". . .additions to the Program which (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program."[55] So licensees can avoid the copyleft application of the CPL by both creating a separate module of software and ensuring that separate module of software is not a derivative work of the code brought in under the CPL.

Like the MPL, the CPL includes patent defense clauses that serve to terminate the patent license granted in the event that a Recipient brings certain kinds of patent litigation. The first of these has broad application that potential licensees should consider carefully and that may deter them from accepting software under the CPL at all: it terminates any patent licenses granted by a Contributor to the Recipient under the CPL as of the date of filing the litigation if the Recipient brings litigation against the Contributor "with respect to a patent applicable to software (including a cross-claim or counterclaim)".[56] So, as with the MPL patent defense clause, the CPL provides that any litigation relating to a software patent, regardless of whether it applies to the licensed software, leads to termination of the patent license.

The second of the CPL patent defense clauses applies to litigation "against any entity" and "alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes" the Recipient's patent(s). Note that these termination provisions apply only to patent licenses granted; so if a Contributor has not granted any patent licenses in connection with its use of the CPL, no license termination would occur.[57]

Another noteworthy provision of the CPL is its Section 4, which covers the indemnity and other responsibilities of commercial software distributors. Section 4 states that Contributors that include the Program in commercial product offerings "should do so in a manner which does not create potential liability for other Contributors." This Section goes on to provide that any Contributor that includes the program in a commercial product offering (a "Commercial Contributor") agrees to "defend and indemnify" every other Contributor against losses incurred by the other Contributor "to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering."[58] However, this indemnity obligation does not apply to infringement claims, and is therefore more likely to arise in a context where a licensee offers warranties or other performance claims relating to the commercial product. Also, licensees should keep in mind that this indemnity provision only applies to commercial distributions, and only with respect to the "acts and omissions" of the Commercial Contributor, so such entities may be able to

[55] Common Public License v. 1.0, Section 1, available at http://opensource.org/licenses/cpl1.0.php
[56] Common Public License v. 1.0, Section 7, available at http://opensource.org/licenses/cpl1.0.php
[57] Common Public License v. 1.0, Section 7, available at http://opensource.org/licenses/cpl1.0.php
[58] Common Public License v. 1.0, Section 4, available at http://opensource.org/licenses/cpl1.0.php

avoid the indemnity obligations if they are not at fault. Nonetheless, commercial licensees need to evaluate their potential exposure with respect to indemnification of all Contributors before distributing CPLed software in a commercial context.

MANAGING FOSS IN THE ENTERPRISE

> *. . .Admit that the waters around you have grown*
> *And accept it that soon you'll be drenched to the bone. . .*
> *You better start swimmin' or you'll sink like a stone. . .*[59]

FOSS is becoming increasingly pervasive, and any entity that relies on technology for its business needs to "start swimmin'" in one way or another. But, other than acknowledging the nature and increasing influence of FOSS, and getting some familiarity with FOSS license terms, what should a company do in order to manage FOSS in its enterprise and to reduce infringement and other risks associated with FOSS? Rather than advising companies to panic and run away screaming, we believe that companies likely can improve their businesses and avoid undue risk if they "swim with the tide" when it comes to managing FOSS in their enterprises.

"Swimming with the tide" doesn't necessarily mean converting one's business or technology to one entirely FOSS-based. Nor does it mean that companies must embrace FOSS wholeheartedly by incorporating FOSS code into their products or becoming the next major donor to the FSF. While engaging more deeply with FOSS might well be the right approach from a business and profit perspective, a sensible response to FOSS might involve simply leveraging existing policies and practices around when and how code is brought into the organization, and carefully monitoring third party license requirements throughout the technology development and marketing process. Many companies already have such policies and practices in place when it comes to proprietary technology they acquire. So working with FOSS could involve nothing more than thoughtfully modifying those policies and practices so as to apply them to FOSS code.

FOSS Policies and "Best Practices"

In developing any policies and practices around FOSS, it is critical that companies recognize that, with the growth of FOSS, has come a fundamental shift in how technology is developed and acquired. That shift requires a change in how companies manage their intellectual property assets, including the acquisition and distribution of those assets. It also necessitates a change in the role of functional groups that support company business – including the role of the legal function in technology acquisition and distribution. Not that long ago, lawyers used to be company "gatekeepers" without whose approval no software could come in the door. These lawyers, presumably, were in close

[59] Bob Dylan; *The Times They Are-a Changin'*

touch with company managers about purchasing decisions. So incoming technology went through a technical, legal and business review before it entered the company and nothing came in without requisite company authorization. If you wanted to know what third party code was lurking in company products, you could go and look at the legal files. With FOSS, this "gatekeeping" ability has gone away. Now, hundreds of thousands of software programs are freely available for download by anyone with internet access, and companies are dependent upon the knowledge and discipline of their engineering staffs to screen incoming code and associated licenses. This dependence increases the importance of communication and good relationships among those engineering teams and other company functions if intellectual property management policies are to be developed and followed. Therefore, taking steps to enhance cooperation and communication between the company engineering, legal and other departments becomes a critical part of any FOSS management strategy.

Just as with any proprietary code they acquire, commercial entities ought to be thinking about how FOSS code fits within their overall business plan and strategy, future product line, distribution and revenue model. Once these questions have been answered, companies should then carefully select those FOSS or proprietary vendor products required or most useful to implement that particular product development and marketing strategy. Companies should be doing a technical review and evaluation of potential incoming third party code for suitability within their own product; and reviewing (and/or negotiating) the applicable licenses to be sure that they have the rights they need throughout the evaluation, development, production and distribution processes. Every company should have systems in place to monitor and enforce compliance with those license terms, and to track software acquisition and development, throughout all phases of technology development, production and release.

Checking the Source of the Source.

As part of the analysis described above, companies seeking to manage their intellectual property assets in the age of FOSS should ensure that the technical and business review that they conduct with respect to any and all incoming code covers such questions as:

- What is the "pedigree" of the code?
    - Who wrote it and can the company confirm its origin?
    - Does it contain any third party code?
        - If so, has the origin of that code been established?
        - Have all necessary third party rights been transferred?
        - What license terms apply to that third party code?
- What is the code quality – has it been screened and scrubbed? Has the company done a complete technical review?

Note that, had Linksys and Cisco pursued this kind of analysis in more depth "upstream", they might have avoided the source code release nightmare described earlier in this paper. A "take-away" from the Linksys case is:

> . . . the difficulty of doing enough diligence on software development in an age of vertical dis-integration. Cisco knew nothing about the problem, despite presumably having done intellectual property diligence on Linksys before it bought the company. But to confound matters, Linksys probably knew nothing of the problem either, because Linksys has been buying the culprit chipsets from Broadcom. . .and Broadcom also presumably did not know, because it in turn had outsourced the development of the firmware for the chipset to an overseas developer.[60]

The Linksys example highlights the importance of companies' establishing and diligently maintaining processes for monitoring third party license terms and compliance with those terms throughout the software development and production process, including the evaluation/testing, development, production and distribution processes. In particular, companies need to be sure they have adequate and necessary rights to any "downstream" licensing and redistribution. They also should be sure they have confirmed any necessary rights to offer technology on a service provider ("ASP") or other basis. And companies need to be alert to FOSS licensing requirements, and particularly the "copyleft" provisions of any licenses governing incoming code, throughout all stages of software development and distribution.

These kinds of "best practices" are explored further in other papers and will not be delved into in more detail here.[61] Suffice it to say that companies seeking to foster not only optimal technical development, but also compliance with the rights they have been granted by third party licensors, must be vigilant about implementing such processes. If companies are alert to the issues presented by bringing FOSS into their enterprise, and manage its implementation thoughtfully, they can avoid unnecessary risks. And, given the potential that FOSS offers for developing innovative technology "more, better, faster", they may also find that opening up their enterprise to FOSS is a sound business strategy.

[60] Meeker, "The Legend of Linksys" in LinuxInsider (January 1, 2006), available at http://www.linuxinsider.com/story/47987.html

[61] See Murray; "*Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise*" (October 2005) pp. 3-4 available at http://www.mataulegal.com/articles.htm

