Living With Open Source: Implementing, Managing And Enforcing A Uniform Policy For Your Enterprise

By Andrew T. Pham and Matthew B. Weinstein

The explosion of open source software and its associated licenses has dramatically increased the complexity of deciding how to create, manage and enforce a uniform open source policy for *"closed-source"* commercial enterprises.[1] Because of the complexity and risks associated with open source—where source code is made freely available for all to review, edit, and use—many closed-source commercial enterprises discourage or prohibit use of open source; a common and short-sighted practice. Open source can be an invaluable tool, and its risks can be understood, managed and controlled. This article aims to assist engineers, developers, managers, licensing professionals, in-house counsel and other open source practitioners in developing a consistent and effective open source policy to enable a peaceful coexistence between open source and closed-source in the competitive global marketplace.

Free and open source software ("FOSS" or collectively "open source") has firmly established itself as part of the fabric of modern software development with over 180,000 existing software projects, tens of billions of lines of code, and more than 1,400 unique licenses.[2] FOSS is not only made available to be used by other developers, but in most instances, copied, modified and distributed by them as well. It is typically offered without royalty requirements and distributed for both commercial and not-for-profit use, depending upon the wishes of its creators. Unlike commercial licenses, which are commonly negotiated on an individual project basis, use of standardized FOSS licenses by developers often leads to confusion. Although open source is *"free"* it is still in nearly all cases *"proprietary,"* meaning that someone still owns the source code and deserves credit for its development and subsequent uses. Open source licenses therefore generally contain instructions on how to give attribution in distributed works. "Free" does not mean *public domain* software, so enterprises should place the same care in FOSS as they would putting any third party code into their product.

■ Andrew T. Pham,
Verint Systems, Inc.,
Associate General Counsel,
Global IP,
Santa Clara, CA, USA
E-Mail: andy.pham@verint.com

■ Matthew B. Weinstein,
Dickstein Shapiro LLP,
Associate,
Washington, DC , USA
E-mail: WeinsteinM@DicksteinShapiro.com

Open source code is also beneficial to company projects in terms of cost savings and speed. Therefore, it is paramount that in software development, companies develop and implement uniform FOSS policies and guidelines for their enterprises. As discussed in this article, this is because using FOSS-derived code can carry little or no risk—when the code is paired with an academic-style license such as Berkeley Software Distribution (BSD)—or critically high risk, such as when code is licensed under the GNU Public License (GPL). These policies ensure FOSS is used properly and legally, FOSS creators are given credit for their work, and liability to the enterprise is reduced. This article explains such a policy and associated processes and controls, enabling safe acceptance and peaceful coexistence of open source in commercial projects.

To balance the needs of all stakeholders (legal, business and engineering) the goals of such a policy should be centered on three central themes:

• **Protecting The Enterprise's Intellectual Property Rights and Source Code:** While valuing open source, companies can still protect the commercial value of company close-source code, trade secrets, patents, copyrights and trademarks by classifying FOSS into buckets or classes based on license restrictions. When considering whether to use open source code in a project, personnel must remember that such use could force the company to release the entire project as open source under the same license and thus lose any commercial viability (and patent rights). This problem is mitigated

Special thanks to Larry Aaronson, Peter Fante, Jon Grossman, Brad Olson and Steve Weisburd for their support and contributions.

1. For the sake of simplicity, "open source" as used in this article shall mean software source code publicly available for use and review, whereas "closed-source" shall mean software source code is kept confidential.

2. See, e.g., *http://www.sourceforge.net*, which has the largest repository of open source codes and applications on the Internet.

and controlled by using the correct type of FOSS or finding alternate code through design or commercial licensing.

- **Minimizing Potential Liability To The Enterprise:** Most open source licenses have disclosure requirements triggered by the further distribution of that code. In short, incorporating open source forces the company to assume legal liability for that code and this must be done only with the utmost care. Companies can reduce risk by learning, understanding, creating and applying company policies, guidelines, and procedures when dealing with open source code or products. This includes proper attribution and also guaranteeing the software works properly by testing FOSS before use to reduce liability to the enterprise.
- **Promoting Efficient Software Development:** An enterprise should consider using FOSS when it can be used to make software development more efficient, however without impacting the company's intellectual property rights or unduly increasing liability. Using FOSS can greatly speed up projects when using industry standard software and technologies (e.g., utilities, tools, libraries, etc.).

Executive management needs to recognize that the foundation of any policy for FOSS within the enterprise to meet these goals should revolve around a classification of FOSS license terms.[3] Consider creating and implementing a FOSS term classification system for your enterprise by dividing licenses into categories based on their usability, dividing FOSS software by licenses that are permissible for use without the "same" license restriction, and licenses which must be reviewed in more detail because of restrictions in the way a new project can be licensed if used externally.[4] Since the latter may not allow commercial use, it is also important to decide which groups of stakeholders within a given enterprise are necessary to implement such policies, and what their roles should be.

I. FOSS And The Modern Enterprise

Open source is a valuable tool, but one that must be understood to be used effectively. At its core is the concept that expressive works can be better utilized and developed by allowing open access to the source code. A commonly accepted theory is that having more eyes looking at the code will result in faster discovery and correction of errors than the traditional closed-source code system.[5] Although "Free"[6] and "Open Source"[7] software are not necessarily co-extensive, for legal purposes their integration into the corporate environment is similar. Both are designed to encourage free distribution of software and availability of software source code.

FOSS is designed and distributed in much the same manner as traditional commercial software, however FOSS licenses, instead of restricting rights, expand them by including additional obligations[8] in five important and otherwise traditionally restricted areas:

- Redistribution (typically, the license is royalty-free);
- Access to code (typically, all who distribute must allow free access);
- Derivative works (typically, the FOSS license "trickles down");
- Discrimination (typically, no one may be barred from using the software or code); and

3. See, e.g., Report of License Proliferation Committee and draft FAQ (categorizing all the currently approved Open Source Initiative (OSI) licenses into three categories: those that are popular and widely used or with strong communities; special purpose licenses; and those licenses that are redundant, which includes those that are non-reusable and other miscellaneous licenses), *http://opensource.org/proliferation-report* (last visited December 29, 2008).

4. See, e.g., Free Software Foundation Licenses, *http://www.fsf.org/licensing/licenses/index_html* (last visited December 29, 2008) (categorizing open source licenses based on: "Whether it qualifies as a free software license. Whether it is a copyleft license. Whether it is compatible with the GNU GPL…Whether it causes any particular practical problems").

5. The reverse could also be argued–that is, FOSS may not be supported like other third party commercial code.

6. The Free Software Foundation provides an apt definition. "Free software is a matter of liberty, not price. To understand the concept, you should think of 'free' as in 'free speech,' not as in 'free beer.'" See *The Free Software Definition*, *http://www.fsf.org/licensing/essays/free-sw.html* (last visited December 8, 2008).

7. "Open source doesn't just mean access to the source code… open-source software must comply with the following criteria: 1. Free Redistribution… 2. [Availability of] Source Code… 3. [Allow] Derived Works… 4. [Protect] Integrity of The Author's Source Code… 5. No Discrimination Against Persons or Groups… 6. No Discrimination Against Fields of Endeavor… 7. [Free] Distribution of License… 8. License Must Not Be Specific to a Product… [i.e., the rights attached to the program must not depend on the program's being part of a particular software distribution]… 9. License Must Not Restrict Other Software… [and] 10. License Must Be Technology-Neutral… [i.e., no provision of the license may be predicated on any individual technology or style of interface]" *The Open Source Definition, http://www.opensource.org/docs/osd* (last visited December 8, 2008). See also ABA Section of Intellectual Property Law, *An Overview* of *Open Source Software Licenses, http://www.abanet.org/intelprop/opensource.html* (last visited December 1, 2008).

8. Stephan Fishman, *Open Source Licenses Are Not All the Same, http://www.onlamp.com/pub/a/onlamp/ 2004/11/18/licenses.html* (last visited December 1, 2008).

- Closed-Source Restriction (typically, the license prohibits a non-open source distribution).

In general, FOSS has demonstrable economic benefits. FOSS encourages freedom of creativity and expression in development; it gives a company the opportunity to build, from the ground up, a user community as well as developer community; and creates an open platform for collaboration and communication. Use of FOSS can also save often overwhelming development costs when working with industry-common features or frameworks. The careful management and integration of FOSS may enable some enterprises to implement a hybrid development model, where products are created from the combination of internally-developed and open source codes (and other third party codes or technologies).

Although there are many tangible benefits to integrating FOSS into a software platform, it is important to be mindful of potential pitfalls that can have serious legal consequences. These pitfalls can often interfere with open source's peaceful coexistence with closed-source and commercial development.

Remember that under the terms of certain licenses, incorporating open source code into a software project could force a company to release an entire project to the public as open source under the same license. That result could be devastating to the project's commercial value and to the company. Even under FOSS licenses with friendlier permissive terms, without following the proper procedures, a development project might unknowingly incorporate code that infringes the claims of another's patent. Additionally, most FOSS licenses have disclosure requirements triggered by the further distribution of that code. Lastly, many FOSS licenses do not include any intellectual property representations, warranties or indemnities. In short, incorporating open source software may unwittingly force a company to assume liability for that code and such decision to incorporate FOSS software must be done with the utmost care under a uniform policy and process.

II. Why Is A FOSS Policy So Important?

Companies invest extensive time, money and resources in the development of their products. Yet many companies have already included FOSS-licensed code in their products without fully comprehending the slumbering risks of having done so.[9] Thus, ensuring this investment is protected is as important to the overall success of the company as the ability to sell its product to customers. Issues raised by open source development and licensing may encompass many different bodies of law, but there is now existing, and ever-expanding U.S. case law on FOSS which has confirmed that FOSS licenses are enforceable and will be upheld by U.S. courts.

In a recent case before the United States Court of Appeals for the Federal Circuit, *Jacobsen v. Katzer*,[10] the Court held that open source license conditions are enforceable under the U.S. copyright laws. Jacobsen's open source license at issue allowed anyone to use his software so long as his conditions were met (such as making any modified code freely available). The district court held that violation of the open source conditions would have to be remedied through a contract claim rather than under U.S. copyright law.

The Federal Circuit, however, sided with the copyright holder—holding that "[c]opyright holders who engage in open source licensing have the right to control the modification and distribution of copyrighted material."[11] In a pronouncement that crystallized the issue of the enforceability of FOSS licenses, the appellate court stated that:

> '[U]nauthorized editing of the underlying work, if proven, would constitute an infringement of the copyright in that work similar to any other use of a work that exceeded the license granted by the proprietor of the copyright.' Copyright licenses are designed to support the right to exclude; money damages alone do not support or enforce that right. The choice to exact consideration in the form of compliance with the open source requirements of disclosure and explanation of changes, rather than as a dollar-denominated fee, is entitled to no less legal recognition. Indeed, because a calculation of damages is inherently speculative, these types of license restrictions might well be rendered meaningless absent the ability to enforce through injunctive relief.
>
> . . . .
>
> The clear language of the Artistic License creates conditions to protect the economic rights at issue in the granting of a public license. These conditions govern the rights to modify and distribute the computer programs and files included in the downloadable software package. *The attribution*

9. William H. Venema, *Open Source Software*, National Law Journal, October 20, 2008, available at *http://www.ebglaw.com/files/24006_00510080013EpsteinB.pdf* (last visited November 1, 2008) (discussing that "[m]any executives whose enterprises routinely use open source software fail to understand that any violation of the applicable open-source license is a violation of copyright law, which can give rise to liability for significant damages").

10. 535 F.3d 1373 (Fed. Cir. 2008).

11. *Id.* at 1381-82.

and modification transparency requirements directly serve to drive traffic to the open source incubation page and to inform downstream users of the project, which is a significant economic goal of the copyright holder that the law will enforce. Through this controlled spread of information, the copyright holder gains creative collaborators to the open source project; by requiring that changes made by downstream users be visible to the copyright holder and others, the copyright holder learns about the uses for his software and gains others' knowledge that can be used to advance future software releases.[12]

The danger of FOSS enforcement and litigation is real. For example, on December 11, 2008, the Free Software Foundation filed a lawsuit against Cisco Systems alleging violations of the GNU General Public (GPL) and Lesser General Public Licenses (LGPL) in its Linksys line of products.[13] The Software Freedom Law Center (SFLC) has also filed and settled multiple suits on BusyBox, an open source software licensed under the GNU GPL version 2.[14] Moreover, FOSS licenses have also been upheld and found enforceable worldwide. In Germany, courts have enjoined product distribution,[15] awarded legal fees,[16] and copyright infringement damages for FOSS license violations. Suffice it to say, FOSS issues are a daily component of the software development landscape.

III. How To Implement And Define A Corporate FOSS Policy

FOSS is now a firmly entrenched part of the software development community; there are literally hundreds of open source license agreements. Some are home-grown while others are modeled after mainstream FOSS licenses such as BSD, Apache, MIT-style, or GNU GPL. FOSS is widely used, often without the company legal department's knowledge.[17] Because a blanket prohibition is nearly impossible to enforce, it is very important to understand the pros and cons of the different aspects of FOSS. Because of the many risks detailed above, compliance with the policy is necessary and a company should consider setting strict penalties, including potential dismissal, for failure to comply. Open source risks cannot be eliminated, but they can be contained and controlled, allowing for peaceful coexistence with closed-source projects. This is possible by adopting a framework for classification of FOSS licenses as a foundation for a FOSS policy.

While there are many different ways to categorize FOSS licenses, arguably, the greatest risk posed by FOSS to commercial enterprises is the so-called 'copyleft' provisions. Copyleft—a portmanteau of the words *copyright* and *left*—refers to a general method or licensing scheme for making a program or other work open source, and requiring all modified and extended versions of the program to be open source as well. These are sometimes referred to as 'viral' or 'reciprocal' licenses because any works derived from a copyleft work must themselves be copyleft when distributed.

Generally speaking, critical terms in copyleft license are:

- Those affecting a developer's rights to copy, modify, and use,[18] and
- Those requiring, if the software containing a copyleft license is "integrated" with your enterprise's software, that you grant a license to your enterprise software under the terms of the same FOSS license and the requirement that you enterprise source code be made available at no charge.

It is important to note that copyleft-based open source licenses have different strengths and effect on derivative works. For example, "weak-to-medium" copyleft licenses are generally used to license FOSS libraries. In programming-speak, these FOSS libraries are "*linked*,"[19] or compiled to closed-source codes. Generally, "*dynamically*"[20] linking to these FOSS

12. *Id.* at 1381-83 (citing *Gilliam v. ABC*, 538 F.2d 14, 21 (2d Cir.1976)) (emphasis added).

13. Free Software Foundation News, *Free Software Foundation Files Suit Against Cisco For GPL Violations*, *http://www.fsf.org/news/2008-12-cisco-suit* (last visited December 28, 2008).

14. For example, see Software Freedom Law Center News, *SFLC Files GPL Violation Lawsuit Against Extreme Networks, Inc., http://www.softwarefreedom.org/news/2008/jul/21/busybox/; BusyBox Developers Settle Case With Extreme Networks, http://www.softwarefreedom.org/news/2008/oct/06/busybox-extreme-settle/* (last visited December 28, 2008).

15. See Jorge Contreras, Jr. & Belinda M. Juran, *Second Injunction Enforcing GPL Issued in Germany, http://www.wilmerhale.com/publications/whPubsDetail.aspx?publication=346* (last visited December 9, 2008).

16. See Mayank Sharma, *GPL passes acid test in German court, http://www.linux.com/articles/57353* (last visited December 9, 2008).

17. Venema, *supra note 6*.

18. See GNU General Public License v. 3.0, *http://www.gnu.org/licenses/gpl-3.0.txt* ("adapt[ing] all or part of the work in a fashion requiring copyright permission…must license the entire work, as a whole, under this License").

19. A program may consist of many source code files that are complied into object code to form the program and linked to other object code to form an executable program.

20. Dynamic linking means that modules, such as libraries, are loaded into an application program at runtime, rather than being linked in at compile time, and remain as separate files on disk.

libraries and then commercially redistributing will not trigger the requirement for the combined FOSS and closed-source work to be distributed under that FOSS library's license. Only changes to the FOSS library itself become subject to the copyleft provisions of the copyleft license (not changes to the closed-source software that links to the FOSS library). Further, copyleft-based open source licenses can be examined for their stances on patent protection, e.g., whether a license promotes patent protection, or takes a negative view of it (such as the non-assertion clauses present in GPL v3.0).[21]

In view of the above, one can classify FOSS licenses based on the copyleft provision as follows (e.g., based on the existence and obligations imposed by the copyleft provision or the lack thereof):[22]

Class A. FOSS components licensed under Class A licenses should be allowed to be used with minimal consultation with the legal department, but prior review by engineering management is highly recommended.[23]

Class B. FOSS components licensed under Class B licenses can also be considered for use in company projects under the appropriate conditions.

Class C. FOSS components licensed under Class C licenses should not be used in company projects or should be strictly monitored and controlled.

This classification scheme will help closed-source commercial enterprises manage and mitigate risks associated with unauthorized and unmonitored use of FOSS that could jeopardize the commercial value of the company's products because of undesirable FOSS license terms (in most cases, the copyleft provision).

A. Classifying FOSS Licenses

The proposed herein classification system looks at how an enterprise can use a FOSS license based on its restrictions. Some FOSS licenses impose the "same license" requirement (i.e., the open source license) on the distribution of derivative works, while others allow derivative works to be released commercially—that is, distribute the open source code and modifications therein as closed-source (e.g., in object code[24] format only). So companies should look at the extent to which a license requires derivative works to use the same open source license. More commonly-used and easily classified FOSS licenses are discussed below, but other licenses will need to be classified according to a policy implemented in the given enterprise. Further, when new FOSS licenses or components are introduced, they should likewise be examined and assigned to a class.

The first consideration of FOSS is the end use. A large concern with using FOSS is often how a commercial project will be affected, but the concern may be lessened if the FOSS is not for an externally released or distributed product, as a "same" license requirement is triggered upon distribution of the derived work.

Consider the following test when reviewing a FOSS component, its license and usage within a commercial project.

1. Are you creating derivative work? Has source code been modified, adapted or combined with other codes? If not, the FOSS component can likely be used internally and distributed without too much concern over the copyleft terms.

2. Are you distributing the derivative work? If an enterprise is planning to distribute a derivative

21. See GNU General Public License v. 3.0 § 11 (stating "[i]f, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it").

22. A category-based scheme was proposed in 2006 by Sun Microsystems. See Simon Phipps, *Free and Open Source Licensing, http://mediacast.sun.com/users/sunmink/media/SunLicensingWhitePaper042006.pdf* (last visited December 9, 2008) (discussing Sun Microsystem's approach to free and open source licensing, in that it views the FOSS continuum as a "virtuous cycle, or an endless circle in which developers share a source code commons, create derivative works from that code, and then contribute back to the commons in the form of innovation… Within this cycle, there are various degrees of freedom with regard to how developers structure their code and license the derivative works they create from the code commons. The degrees of freedom that characterize the cycle shape the community that forms around the use of code and reflect the kind of licensing that is associated with that use").

23. Note that use of the open source code within the company for research and development purposes (e.g., software development tools or testing tools) may be distinguished from use of open source code within a company project for outside distribution. Depending on the terms of the particular license, if code is never distributed, it may not need to be licensed or governed by the specific terms of the open source license. But from a legal review perspective, there should be no distinction between FOSS used in internal research and development (or testing) and FOSS used in commercial products.

24. Object codes are codes that are produced by a compiler from the source code, usually in the form of machine language that a computer can execute directly. In other words, a compiler is a specialized program that converts source code into object code, usually a *machine code*, which can be understood directly by a computer.

work, then the FOSS license must be reviewed to identify the copyleft provision, if any.

3. Does the FOSS license contain any copyleft provisions? If yes, the analysis moves to determine the strength of the viral effect of the copyleft provision and what source code disclosure requirement is present.

4. Are you required to disclose the source code and license any new code under the same conditions for the derived file? One way to view such a requirement is that certain "chapters" or files of the code base must be disclosed, but not all of the files or the entire project or program. In this case, only the source code to specific file where the FOSS component is modified, adapted, inserted or combined are at risk.

5. Are you required to disclose the source code and license any new code under the same conditions for the derived project (e.g., the derived file plus any other files that are "combined" with the derived file)? This requirement is akin to forcing disclosure of the entire "book," project or program. While the disclosure of source code to a single file may be acceptable to some companies, the disclosure of source code to the entire project or product would be unacceptable (and devastating) to nearly every closed-source commercial enterprise.

Based on the above tests, one can at least classify FOSS licenses into three categories.

Class A—Non-Copyleft Licenses. Sometimes called 'Non-viral' or 'Academic' licenses, these types of licenses are preferred for commercial use as they do not place "same" license requirements on derivative works (works developed with FOSS code.) Some examples are BSD, MIT, and Apache licenses.[25]

This class of non-copyleft licenses allows companies to choose the license for the new work[26] (which means that a company can release the derived work as closed-source and charge a royalty for them). These licenses may, however, impose other non-restrictive conditions pertaining to attribution (how credit is given to the creator). There may also be a release of liability requirement—meaning that the users must agree not to file suit against the original author, or a notice of modifications requirement if original code has been modified.[27] These licenses should be screened before use and distribution, despite the fact that using Class A code may not pose any major issues for a commercial developer.

Class B—Bounded Copyleft Licenses. FOSS licenses with copyleft terms should be considered carefully before use since they usually require new codes to be licensed under the same license.[28] Not all copyleft-based licenses impose the "same" license requirement on files that do not contain code from the original open source code; those files can be licensed in any license.[29] This enables companies to possibly use these FOSS components commercially, although the terms under which they are licensed are definitely considered 'copyleft.' Examples include GNU LGPL and Microsoft's Public License (MPL).[30]

Based on the above assumptions, these Class B "bounded" copyleft licenses can be thought of as a "file-based" licenses, meaning the viral effect of the copyleft provision only affect the file in which the FOSS component is used, modified, incorporated, combined, etc. In other words, the same license requirement is determined by the file or module in which the derivative work is contained—the license is 'bounded.'[31] To an extent, these licenses allow developers to use FOSS code in a non-limiting manner.[32] So succinctly, a newly-created file is subject to the license requirements if it is a modification of an existing FOSS file.[33] On the other hand, newly (and independently) developed files are not subject to the requirement.[34]

Class C—Unbounded Copyleft Licenses. This class of FOSS licenses can be the most problematic to use commercially because these licenses require that *all* combined files—even those not containing FOSS code at all—must be licensed under the same license as the FOSS project.[35] This class of "unbounded" copyleft licenses presumes that the derivative works will become part of a compiled program.[36] Based on this assumption, the licenses require that any new code (modified or not) from the FOSS project is required to use the same license.[37] This "unbounded

25. Simon Phipps, *Free and Open Source Licensing* at 7, *http://mediacast.sun.com/users/sunmink/media/ SunLicensingWhitePaper042006.pdf* (last visited December 9, 2008).

26. Id. at 4.

27. Id.

28 Id. at 4-5.

29. Id.

30. Id. at 7.

31. Id. at 4-5.

32. Id.

33. Id.

34. Id.

35. Id.

36. Id.

37. Id. at 5.

same" license feature is where problems come in for enterprises using FOSS for projects that will be released externally or commercially. Philosophically, supporters of this class of strong, unbounded copyleft licenses included this viral requirement to promote the availability of free software and to allow licensed projects to become sources of additional, same-licensed FOSS code.[38] Examples of this include GNU GPL (Version 2 and 3), Common Public License, and Open Public License.[39]

B. Leveraging The Classification Scheme To Create A Uniform Policy

Not all FOSS licenses should be automatically discounted for providing value to an enterprise, and an open source policy that allows the use of FOSS components having commercially-friendly or commercially-adaptable FOSS licenses is therefore ideal.[40,41] Generally speaking, non-copyleft licenses or perhaps even "weak-to-medium" or bounded copyleft licenses where use is monitored and controlled may be acceptable in closed-source commercial projects.

The fact that FOSS is often referred to as "Open Source" or "Free Software" can complicate the use of FOSS components in commercial development. Since licensees can use "Open Source" software to access, use, copy, modify and distribute without making royalty payments to the licensor, and combine this open source software with other software code, it is better to think of open source as "commercial-leaning" rather than free. If the license is unbounded copyleft it will not allow new commercial distribution, but, if the license granted with the open source software is academic, nonviral or permissive, the licensee can take the modifications to the FOSS code as private or "closed-source." Hence, Class A-licensed open source software is the ideal FOSS for commercial use. With the appropriate controls and process in place, Class B bounded copyleft licenses may also be used.

To implement such a policy using the classes outlined herein, companies should assess the current state of their FOSS usage and involve team members from all areas of legal and engineering. In designing the policy, decide who the stakeholders are, what FOSS is currently used in the company, how to manage future uses of FOSS and what steps should be undertaken for each FOSS review and approval.

1. Stakeholders

The success of a uniform FOSS policy and use of FOSS is greatly affected by stakeholders, which is why it is important to know who is involved and what they do. All, including the engineers, the managers, and the legal counsels, should be invested in the process to ensure the success of the FOSS policy, process and implementation.

The Engineering Department should be vested in the design process and carefully consider the consequences of all FOSS uses within their projects. One critical task is to seek guidance from other departments, and most importantly, the legal department, to ensure the FOSS code does not infringe upon others' intellectual property rights among other considerations noted above. Engineers and developers should thoroughly investigate any particular open source project being considered, including its longevity, number of contributors, security problems, if any, and its method of releasing "bug" fixes, contributions and updates.

Management should monitor, track and, as appropriate, approve all FOSS uses within their projects. Engineering and product managers need to ensure that any open source use respects the terms of the license governing the open source project (e.g. including attribution language, providing copies of or references to locations of the license agreement and source code). All managers must understand the critical need to protect the company's intellectual property rights and source code (perhaps even more critical than patent rights), seeking guidance from the legal department where necessary. It is important to note that product managers must have a role in the FOSS review and approval process. Product managers can often take the lead role by planning for FOSS issues in their product roadmap or specification requirement documentations.

The Legal Department should be the gatekeeper for approving all uses open source code and FOSS licenses. The legal department should consider the licenses governing the FOSS code and, to a reasonable degree, make an assessment as to whether the FOSS code potentially infringes any intellectual property rights of others. The legal department should also work with engineering and product managers

---

38. Id.

39. Id. at 7.

40. Note that the use of open source tools in the creation of software (as opposed to the incorporation of open source code into a commercialized product) may not create any obligation to release any code created by the open source tool as open source. Regardless of the end use, it is good practice to check all open source licenses to ensure that there are no unintended consequences.

41. Simon Phipps, *Free and Open Source Licensing, http://mediacast.sun.com/users/sunmink/media/SunLicensingWhitePaper042006.pdf* (last visited December 9, 2008).

to develop a process to request, review, approve, and track all FOSS components. Finally, train, train, train! The legal department should take the lead in providing FOSS training and awareness. Teach your managers and developers to selectively fish (and it is prudent to be selective with FOSS) and you will feed them for a lifetime.

2. How To Deal With FOSS Currently In Use

In today's global development ecosystem, open source may have already become a part of your enterprise's development cycles. Many FOSS components are so widely used in commercial software that they are almost considered de facto industry standards. For example, the zlib data compression library, the OpenSSL secure sockets layer library, and the Boost C++ libraries are all well-known and widely-adopted open source implementations of commonly—used standards and technologies. In fact, the research firm Gartner predicts that 80 percent of all commercial software will include elements of open-source technology by 2012.[42] Therefore, enterprises should conduct a thorough audit to determine what FOSS, if any, is in use.[43] Documenting all incorporated open source code will greatly help any future audits and transactions (e.g., M&A transactions). Companies can do this by preserving all copyright notices, complying with all attribution requirements and accurately tracking precisely which code is subject to which open source license. This information should include, for example, the code source, its purpose and use for the enterprise, modifications, websites for retrieving updates and attribution requirements. Using this kind of system simplifies the process of checking that the company is complying with open source license terms.

When dealing with FOSS currently in use, always determine if a commercial license is available. Some FOSS components are licensed under a dual-licensing scheme—meaning an enterprise can choose to take a FOSS license or obtain a commercial license for a fee. Another option is to find a similar FOSS with more desirable license terms (e.g., licensed under Class A), or completely rewrite the function or remove the codes from the project if necessary to preserve and protect company intellectual property and source code.

3. Conducting FOSS Reviews

In short, implementing a class-based system like that described above will help you in protecting your company from liability and prevent loss of a project's commercial value due to FOSS license restrictions. Full audits are easily justified as companies without fully-developed FOSS policies are likely to have FOSS issues already present in their code and waiting to surface. Besides an overall audit, however, a discrete review is helpful for each project and should track any proposed FOSS component and which classification it falls under. Developers and engineers should seek approval from their legal department if the FOSS license is copyleft-based, bounded or unbounded.[44] For example, even for FOSS components that are governed by bounded copyleft licenses, control measures must be implemented to track and maintain file-based separation of the FOSS code from the rest of the company's closed-source codes.[45] Unbounded copyleft licenses with same license provisions should be used only in extraordinary circumstances and under tight supervision by the legal department.

Once a FOSS component is identified, determine the specific reason why the given FOSS code is needed, why it is advantageous and consider if the code will be used only within the enterprise or in externally released and possibly commercial projects. Another important consideration involves investigating if the open source project is trustworthy and well supported, and available under an economically reasonable commercial license (which should be preferred by most companies). Whenever FOSS source code is used, engineers should record its use and develop a plan to keep that code updated with security. Finally, all of the open source code should be thoroughly tested before releasing it in a commercial product. Remember, when an enterprise uses FOSS code, it becomes liable as if the FOSS were the company's own code. So engineering must thoroughly test the particular FOSS component before including it in a company project.

Engineers using FOSS should therefore respect all of the license terms. Proper attribution language is also needed for any FOSS code distributed outside the enterprise. Additionally, it may be necessary to include a copy of the relevant open source license agreement, source code, or information (e.g., web address) where that information can be obtained.

42. Gartner Report, *Gartner Highlights Key Predictions for IT Organisations and Users in 2008 and Beyond, http://gartner.com/it/page.jsp?id=593207* (last visited December 28, 2008).

43. A comprehensive audit should include internally-developed codes, outsourced codes (including independent contractors), and third-party codes.

44. *Id*.

45. *Id*.

Lastly, to a reasonable degree, outside legal counsel or the company's legal department should make an assessment as to whether the code potentially infringes any intellectual property rights of others (e.g., others' patent rights).

4. Don't Forget Attribution

An open source policy must address attribution requirements, the lack of which are a common and easily resolved mistake. Since attribution is a condition enforceable via copyright statutes[46] (see *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008)), compliance is absolutely necessary. In virtually all open source licenses, there are obligations imposed by the *attribution provision*, which are very similar to the reproduction of legal notices. This may be a daunting task for some companies. Regardless, attribution requirements are enforceable so companies should consider a process for complying with all terms and conditions of the open source license. To comply with attribution requirements, one suggestion is to break up the effort into three stages: (1) general notice; (2) modest notice; and (3) detailed notice.

A general FOSS notice should provide notice to the end user that certain components of the licensed software may contain FOSS and that, upon request, the licensor can provide the licensee with a list of the FOSS components. It should also make clear that licenses granted by such FOSS components are no less broad than the license granted to the licensee by the licensor.

A "modest" FOSS notice provides for reasonably modest attribution and continues the effort from a general notice. The modest notice should, for example, include the following information as attribution information: (a) Name of the Component; (b) a copyright notice including the name of the Original Developer/Foundation; and (c) a URL to the license.

Sample:

> Ant, Axis, Avalon are each copyright © 2000-2007 The Apache Software Foundation. All rights reserved. Licensed under the Apache License, Version 2.0, which can be obtained at *http://www.apache.org/licenses/LICENSE-2.0.*

A detailed FOSS notice, which follows the specific attribution requirements of each FOSS component and license, is recommended (but may not be practical for some companies). For example, the Apache License, Version 2, requires that the following notice be included in distribution of software containing FOSS components licensed under the Apache license:

> Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at *http://www.apache.org/licenses/LICENSE-2.0*
>
> Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

While the end goal is to provide complete and detailed notices, the above stage process will help enterprises to gradually move toward complete compliance of all terms and conditions of FOSS licenses.

Conclusions And Recommendations

Utilization of FOSS has many advantages and benefits, but challenges to its unauthorized use are increasingly being tested in courts. With this herein proposed uniform policy and system, legal departments can more effectively and uniformly classify FOSS licenses and monitor FOSS usage. Using a class-based policy will be helpful in surveying the plethora of FOSS licenses and help to protect company's intellectual property rights through preventing the use of potentially exposure-oriented FOSS; it can minimize potential liability because it provides for effective legal review of FOSS use; and it will promote efficient software development because it opens the door to approved FOSS-licensed code to be used in commercial development. In the end, open source and closed-source can peacefully co-exist despite their inherent philosophical differences. ■

46. 535 F.3d at 1379 (discussing that "[b]y requiring that users copy and restate the license and attribution information, a copyright holder can ensure that recipients of the redistributed computer code know the identity of the owner as well as the scope of the license granted by the original owner.")