# EASY AS ABC: CATEGORIZING OPEN SOURCE LICENSES[♠]

[♠]Andrew T. Pham[1], Matthew B. Weinstein[2], Jamie L. Ryerson[3]

With more than 180,000 *open source* projects available and its more than 1400 unique licenses, the complexity of deciding how to manage open source usage within "*closed-source*" commercial enterprises have dramatically increased.[4] 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. With a proper open source management framework, open source can be an invaluable resource, and its risks can be understood, managed and controlled. This article proposes a simple, consistent and effective open source *categorization and management system* to enable a peaceful coexistence between open source and closed-source codes.

Free and open source software ("FOSS" or collectively "open source") is a valuable tool, but one that must be understood to be used effectively. The litany of risks associated with use of open source include: having to release a derivative product incorporating open source under the same open source license; incorporating code that infringes a patent; violating an open source license's attribution requirements; and a lack of warranties and indemnities. Given the extensive investment of time, money and resources that goes into product development, it comes as no

[♠] This article is an edited version of the original, which dealt not only with categorizing open source licenses but also a wider array of issues associated with implementing an open source policy. The original article can be found at: Andrew T. Pham and Matthew B. Weinstein, *Living with Open Source: Implementing, Managing and Enforcing a Uniform Policy for your Enterprise*, LES NOUVELLES: JOURNAL OF THE LICENSING EXECUTIVES SOCIETY, March 2010.

[♠] Special thanks to Larry Aaronson, Peter Fante, and Steven I. Weisburd for their support and contributions.

[1] Associate General Counsel, Global Intellectual Property, Verint Systems Inc. Mr. Pham manages all aspects of IP for Verint, including patent strategy, prosecution and litigation, trademark, copyright, licensing and open source.

[2] Associate, Dickstein Shapiro LLP. Mr. Weinstein specializes in patent counseling and enforcement. Before embarking on his legal career, Mr. Weinstein founded and later sold a technology consulting firm that designed and implemented open source software for its clients.

[3] Associate, Dickstein Shapiro LLP. Mr. Ryerson is an associate in Dickstein Shapiro's Intellectual Property Practice Group.

[4] *See* www.sourceforge.net. Also, for the sake of simplicity, the term "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.

surprise that companies may be wary of using open source. Nevertheless, companies that understand the risks can avoid them and be rewarded in terms of cost savings and increased speed to market.

Issues raised by open source development and licensing may encompass many different bodies of law, but there is now existing, and ever-expanding case law on FOSS which has confirmed that FOSS licenses are enforceable and will be upheld in the U.S. and worldwide. To that end, the United States Court of Appeals for the Federal Circuit recently held that open source license conditions are enforceable under U.S. copyright laws.[5] In addition, proponents of open source are actively enforcing FOSS licenses.[6]

Today, FOSS is widely used by software developers, often without the company legal department's knowledge.[7] Because a blanket prohibition is nearly impossible to enforce and the fact that there are hundreds of thousands of open source projects available, it is very important to develop a *scalable system* for managing open source within a closed-source enterprise. Due to the prevalence of FOSS in the software development community, there are literally hundreds of variations of open source license agreements. Some are home-grown while others are modeled after mainstream FOSS licenses such as Berkeley Software Distribution ("BSD"), Apache, MIT-style (Massachusetts Institute of Technology), or GNU General Public License ("GPL"). As such, understanding the different characteristics of open source license agreements – in other words, the basic ABCs or philosophy behind the plethora of open source licenses – is key to the

[5] *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008).

[6] 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. 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. In Germany, courts have enjoined product distribution and awarded legal fees and copyright infringement damages for FOSS license violations. *See* 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 Feb. 15, 2010). *See, e.g.*, Software Freedom Law Center News, *SFLC Files GPL Violation Lawsuit Against Extreme Networks, Inc*., http://www.softwarefreedom.org/news/2008/jul/21/busybox/ (last visited Feb. 15, 2010); *BusyBox Developers Settle Case With Extreme Networks*, http://www.softwarefreedom.org/news/2008/oct/06/busybox-extreme-settle/ (last visited Feb. 15, 2010). *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 Feb. 15, 2010). *See* Mayank Sharma, *GPL passes acid test in German court*, http://www.linux.com/archive/articles/57353 (last visited Feb. 15, 2010).

[7] William H. Venema, *Open Source Software*, National Law Journal, October 20, 2008, *available at* http://www.ebglaw.com/files/24006_00510080013EpsteinB.pdf (last visited Feb. 15, 2010).

success of any open source policy or management framework. Thus, the first step in FOSS management is to develop a framework for classifying or categorizing the thousands of FOSS licenses.

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.

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*,"[8] or compiled to closed-source codes. Generally, "*dynamically*"[9] linking to these FOSS libraries and then commercially redistributing may 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).[10]

[8] 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.

[9] 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.

[10] 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). *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").

In view of the above, one can create the following three-tiered classification scheme into which FOSS licenses may be grouped (typically based on the existence and obligations imposed by the copyleft provision):[11]

- **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.[12]

- **Class B**. FOSS components licensed under Class B licenses can also be considered for use in company projects under 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 simple classification scheme will help closed-source commercial enterprises manage and mitigate risks associated with unauthorized and unmonitored use of FOSS.

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

[11] 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 Feb. 15, 2010) (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").

[12] 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.

as closed-source (*e.g.*, in object code[13] format only). So companies should look at the extent to which a license requires derivative works to use the same open source license.[14]

For example, consider the following when reviewing a FOSS license:

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 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

[13] 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.

[14] The first consideration of FOSS is the end use. Incorporation of FOSS into an externally released or distributed product can pose risks (depending on the type of FOSS license), as "same" license requirements are triggered upon distribution of a derivative work. Obviously, such concerns are mitigated when the use is solely internal.

project or product would be unacceptable (and devastating) to nearly every closed-source commercial enterprise.

Based on the above tests, one can classify FOSS licenses into *a three-tiered classification system*.

**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, Microsoft Permissive License, and Apache licenses.[15]

This class of non-copyleft licenses generally allows companies to choose the license for the new work.[16]. These licenses may, however, impose other non-restrictive conditions pertaining to attribution. 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.[17] These licenses should be screened before use and distribution, despite the fact that using Class A code may not pose any major code-exposure issues for a closed-source enterprise.

**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.[18] 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.[19] 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 Lesser General Public License ("LGPL") and the Microsoft Public License ("MPL").[20]

---

[15] Phipps, *supra* note 9, at 7.

[16] *Id.* at 4.

[17] *Id.*

[18] *Id.* at 4-5.

[19] *Id.*

[20] *Id.* at 7.

Based on the above assumptions, these Class B "bounded" copyleft licenses can be thought of as "file-based" licenses - meaning the viral effect of the copyleft provision only affects 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— thus, the copyleft obligation is 'bounded.'[21] To an extent, these licenses allow developers to use FOSS code in a non-limiting manner.[22] So succinctly, a newly-created file is subject to the license requirements if it is a modification of an existing FOSS file.[23] On the other hand, newly (*and independently*) developed files are *not* subject to the requirement.[24]

**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.[25] This class of "unbounded" copyleft licenses presumes that the derivative works will become part of a compiled program.[26] 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.[27] 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.[28] Examples of this include GNU GPL (Version 2 and 3), Common Public License, and Open Public License.[29]

Based on the above classification scheme, companies can add value by utilizing FOSS components having commercially-friendly or commercially-adaptable FOSS licenses[30,31]

[21] Phipps, *supra* note 9, at 4-5.

[22] *Id.*

[23] *Id.*

[24] *Id.*

[25] *Id.*

[26] *Id.*

[27] Phipps, *supra* note 9, at 5.

[28] *Id.*

[29] *Id.* at 7.

[30] 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.

Generally speaking, non-copyleft licenses or perhaps even "weak-to-medium" or bounded copyleft licenses (if use is monitored and controlled) may be acceptable in closed-source commercial projects. Hence, Class A-licensed open source software may be more acceptable for commercial use by a closed-source enterprise. With the appropriate controls and process in place, Class B bounded copyleft licenses may also be used. Finally, it may be appropriate for closed-source enterprises that are unfamiliar with open source control procedures to limit use of Class C unbounded copyleft licenses until appropriate review and controls measures are put in place.

In conclusion, utilization of FOSS components has many advantages and benefits, but challenges to its unauthorized use are increasingly being tested in courts. With the above FOSS categorization system, legal departments can more effectively and uniformly classify FOSS licenses and monitor FOSS usage – as easy as ABC. Using a class-based policy or management framework 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, with the proper open source control measures and policy, open source and closed-source can peacefully co-exist despite their inherent philosophical (and business model) differences.

[31] Phipps, *supra* note 9.