

# The COMPUTER & INTERNET *Lawyer*

Volume 26 ▲ Number 1 ▲ JANUARY 2009

Ronald L. Johnston, Arnold & Porter, LLP Editor-in-Chief\*

## Categorization of Open Source Licenses: More than Just Semantics

By Gwyn Firth Murray

Since I first started working with open source software, the increase in the amount of software that is “freely available” and in the quantity of associated licenses has been staggering.<sup>1</sup> In 1999, for those of us entering the world of open source licensing, a few licenses seemed to be the only games in town. The most widely used and recognized were the governing license of Linux, the GPL, along with its “lesser” license, the LGPL. Other important and commonly used licenses were the BSD and Apache licenses.<sup>2</sup> Since then, as the number of new open source applications and corresponding licenses have multiplied exponentially, the task of understanding the terms that govern the wide variety of open source licenses has become more of a challenge.

**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](http://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](http://www.open-bar.org)).

During this same period, it has become more critical from a licensee’s perspective to understand and comply with open source license terms. Recent court actions and, most notably, the recent appellate decision in *Jacobson v. Katzer*<sup>3</sup> have supported the enforceability of open source licenses. Organizations such as the Software Freedom Law Center have taken formal action on behalf of open source licensors, with reported success in terms of settlements resulting in the defendants’ publishing applicable source code and payment of financial consideration. Given the increased use and popularity of open source software, more companies are aware of the issues involved in licensing open

Ms. Murray is a graduate of Stanford University Law School and also holds an MA in Latin American Studies from Stanford University. She obtained her BA *magna cum laude* and with distinction in economics from Yale College. Ms. Murray is fluent in Spanish and Portuguese. Ms. Murray gratefully acknowledges the assistance and thoughtful commentary of Anne Firth Murray, Deborah Neville, and Patrick Reilly in preparing this article. Except as specifically noted, views expressed in this article are her own. © Gwyn Firth Murray 2008. All Rights Reserved.



source code, and more potential customers, acquirors, and investors are focusing on licensees' compliance with license terms. Thus, even without the threat of increased litigation, it is more essential than ever that companies using open source software clearly understand the terms on which they bring in that software and how to comply with those terms.

In this context, and as more lawyers and others attempt to understand the growing multitude of open source licenses, numerous attempts to categorize these licenses have been made. Licenses have been grouped into buckets, and terminology has evolved to describe the sub-types of open source software and licensing within those buckets.<sup>4</sup> As time has gone by, some terms have been applied and/or used interchangeably in ways that sometimes confuse, rather than clarify, license terms, thus interfering with licensees' ability to comply with those terms. This article has two purposes—first, to show that *words matter* and, second, to advocate that lawyers, in particular, use more precise terminology when describing open source licensing and licenses.

## Open Source Software Is Ubiquitous

The number of open source projects hosted on *Sourceforge.net* alone as of this writing is greater than 180,000, with the number of registered users listed as approximately 1.9 million.<sup>5</sup> And there are many other repositories from which free and/or open source software can be downloaded, including numerous software repositories hosted by traditionally “commercial” software companies (such as Microsoft, Sun, and Oracle), self-hosted open source projects (such as Apache and PostgreSQL), university and personal Web sites, and for-profit open source companies (such as RedHat and MySQL, the latter recently acquired by Sun Microsystems), among other sources. A number of these companies (including RedHat and MySQL) have built businesses that are based on the use and distribution of open source software; others (such as Sun and IBM) heavily rely on open source software as part of their product strategies; and still others offer services to assist licensees with license compliance (many lawyers, including me, as well as companies such as Black Duck Software and Palamida). The Open Source Initiative (OSI) currently lists approximately 70 OSI-approved licenses,<sup>6</sup> but many more licenses governing the use and distribution of free and open source software are in common use. In addition, estimates show continuing growth in market share for Linux and other open source software. Linux has single-digit overall market share in the operating system market, but this market share is in the double digits on the server/business side. Mozilla's Firefox browser occupies close to 20 percent of that market and shows no signs of slowing down.<sup>7</sup>

This exponential growth in the number of open source software applications and their associated licenses has increased the complexity of license interpretation. Analysis of license terms and compliance requires licensees to determine what each license says within the context of ever-changing technology, whether the various licenses applying to multiple software components are compatible, and how to be sure that they are complying with license terms as they use various components to build complex products. A key point of this article is to urge lawyers and others working with open source software to discuss and portray it objectively and accurately to facilitate license compliance.

## Current Terminology Is Misleading

A lot of “Fear, Uncertainty and Doubt” (FUD) around what open source software is and what it can do has diminished over the past 10 years as open source software has become more commonly used by, and familiar to, the developer and business communities. In the meantime, and as a result of this new ubiquity, a new sort of FUD has been generated stemming from ignorance about, and/or reliance upon, poorly used terminology. As the use of open source software in commercial contexts has grown and as open source software becomes industry-standard, a lot more terms attempting to refer—and define—open source software emerge every day. The plethora of terms used to describe open source software models and licenses is confusing to those attempting to foster licensing compliance. The sheer number of terms and the frequent misuse of some of them seriously hinder the adoption of open source within the enterprise. To remedy this, those of us living in the world of open source licensing should help clarify some of the terminology that is bandied about by those who seek to use and are using free and/or open source software.

Figure 1 shows some of the terms that are used frequently to describe open source technology and licenses nowadays.

## The Problem with Current Terminology

This article will not attempt to define each and all of the terms listed above; rather, it advocates that we agree upon common terms to describe open source licenses and, specifically, that we stop blatant misuse of two terms currently used to mean the opposite of open source: “proprietary” and “commercial.”

## Why Is This Problem Unique to Open Source?

It is true that, in the world of “traditional” or “closed source” software licensing, numerous and varying license terms may apply. Many licenses are negotiated, and terms may differ around how the software is priced

**Figure 1: Terminology of Open Source Technology and Licenses**

Free Software	FSF-Compatible
Open Source Software	Shareware
open source software	Shared Source
Free and Open Source Software	Proprietary
FOSS	Commercial
FLOSS	Non-commercial
Reciprocal	Freely Available Software
Hereditary	Publicly Available Software
Viral	free software
Non-Viral	Academic
Permissive	Restrictive
Copyleft	Not Restrictive
Non-copyleft	Freeware
OSD-Compliant	Adware

(for example, is it on a per-seat or a site-license basis?), whether source code is made available or not, and what

the various promises and obligations of each party are. In these closed source situations, licenses are separately negotiated and vary by product and company. Even shrink wrap or click wrap licenses, which tend not to be negotiated between the parties, contain unique terms that vary depending upon the product and upon the whim and skill of the lawyer(s) who wrote the license.

Most of these “traditional” software licenses are written by, and/or negotiated by, lawyers. Many are drafted using boilerplate language that is used in standard forms or previous agreements and that is familiar to lawyers that have been practicing intellectual property licensing over a number of years. In the world of closed source licensing, the number of people involved in negotiating, drafting, and then interpreting these licenses is limited relative to the general public that has access to open source software and licenses. Those interpreting and building compliance programs around these closed, individually negotiated licenses may be the very same people who wrote the license and who, therefore, are familiar with the particular technology and business goals underlying the license terms.

In the world of open source licensing, however, the situation is different. Although the legal community is increasingly aware of, and involved in, drafting of open source licenses, many of the open source licenses in use

## BRIEF HISTORY OF OPEN SOURCE SOFTWARE

It is key to remember that neither “Free” nor “Open Source” software is new to computer technology. More than 30 years ago, software development was considered part of the scientific enterprise and was developed with much input and sharing among developers and within academic, government, and other research institutions. After AT&T was ordered to divide into Baby Bells, the operating system UNIX, designed to run large mainframe computers, was developed at AT&T Bell Labs, along with “C,” the programming language used to write it. At first, UNIX was licensed by Bell Labs on a non-commercial basis. But then UNIX was forked, and versions of it were licensed on a closed source, royalty basis by companies including IBM and Sun.

In the meantime, Apple Computer and IBM were working on and releasing their personal computers, which required less-cumbersome operating systems than UNIX. Over the course of intervening years, Linus Torvalds wrote Linux, Apple developed its Macintosh operating system, Microsoft developed what is now Microsoft Windows, and IBM, Apple and other companies discovered that they could charge royalties for these closed source software operating systems and applications. This was the emergence of what is now called “traditional” software licensing, which is also frequently (and not always correctly) referred to as “proprietary” or “commercial” software licensing.

All the while, Richard Stallman and other computer scientists were displeased by the closing of UNIX source and by the growth of royalty-based software development and licensing. Stallman founded the GNU Project in order to build an alternative to UNIX, Stallman wrote the original GNU General Public License (GPL), Linus Torvalds agreed that LINUX could become part of the GNU Project, and the GNU Project came under the bailiwick of the Free Software Foundation (FSF). The FSF is the original author and keeper of the GPL (now, as of June 2007, in version 3.0) and the key proponent of the notion of “Free Software.”

# Open Source

---

According to the FSF, the term “Free Software” means that the licensee has the freedom to run, modify and distribute the software. “Free” does not necessarily mean “without cost,” however, in that fees may be charged to cover costs and services related to the distribution of “Free Software.” In order to understand the notion of “Free Software,” one must understand that the freedom of the software runs with the program, not the license. As the FSF has explained, one should think “free speech, not free beer.” [See [www.gnu.org/philosophy/free-sw.html](http://www.gnu.org/philosophy/free-sw.html)].

The “Free Software” movement generally is considered either more encouraging of freedom, or more restrictive of licensees’ freedom, than is the open source movement, depending upon one’s perspective. Under FSF guidelines as to what constitutes “free software,” *all distributed modifications to the licensed software must be licensed under the same terms under which it originally received*. As discussed later, this kind of requirement has been referred to as “copyleft,” “viral,” “reciprocal,” and “hereditary.” And this kind of requirement is what understandably strikes most fear into the hearts of some commercial enterprises. If “viral” open source 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 made available to the general public.

“Open Source Software” (OSS), though essentially identical to “Free Software” for legal purposes, evolved as a result of the 1997 split-off from the “Free Software” movement by members of the open source community who feared that the “viral” characteristics of “Free Software” would limit open source from being used by commercial enterprises and, consequently, would limit the growth and adoption of open source. 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.” [See DiBona, “Introduction” in DiBona, Ockman & Stone (editors), *Open Sources: Voices from the Open Source Revolution*, (O’Reilly 1999) pp.1-17, at 7.] They created the Open Source Initiative (OSI) and also embarked upon a marketing campaign to promote free software. As part of their efforts, they developed and promoted use of a new term to describe the software that they were talking about: “Open Source.” The “Free Software” 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. It is important to understand, however, that these movements grew from different perspectives. “Open Source Software” generally is seen as more commercial-leaning than is “Free 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 “reciprocal” or “copyleft” licenses but, alternatively, may permit the licensee to take his or her modifications to the code private or closed source. These latter sorts of licenses now are being referred to as “academic,” “non-viral,” and “permissive.”

The most common and recognized definition of “Open Source” is the Open Source Definition (OSD), created and maintained by OSI. The OSD clarifies that “[o]pen source doesn’t just mean access to the source code” and requires that a license must comply with 10 specified criteria in order to qualify as an “Open Source Initiative Approved License.” [OSI is now registering this term as a trademark. See <http://opensource.org/trademark>.] Though OSI-certification lends substantial legitimacy to open source licenses within the open source community, the OSD is not the only definition of what constitutes “open source” software. In other words, licenses that are not OSI-approved still may be open source or free licenses in that they apply to software that is available in source code form without requiring payment of associated royalties. An example of such an open source license is the widely used Jason Hunter Servlet license, which makes source code available but restricts the purposes for which the licensed software may be used, particularly in commercial contexts. [See <http://www.servlets.com/cos/license.html>.] Another license that offers code on an open source basis, is the gSOAP Public License, which its authors describe as being “derived from the Mozilla Public License (MPL1.1).” [See <http://www.cs.fsu.edu/~engelen/license.html>.] Neither of these licenses appears on the OSI-approved list.

have been and are written by software engineers and developers. Many of these developer-written licenses are written in technical terms, with the consequence that they are not always easily decipherable by lawyers or civilians who lack a technical background or whose technical background is out of date. Also, some of the more popular and pre-existing open source licenses now are applied to software and technology that has evolved significantly, or is entirely different, from the code and its purpose around which the license originally was written. To evaluate license rights and build an appropriate path to compliance under many open source licenses, lawyers and developers must consider how the particular applications work together, how the programs link to libraries (or for that matter, what a “library” is in terms of software development), how the software is written, and how it is compiled.<sup>8</sup> Particularly when introducing the notion of open source licensing to those new to the field, it is most helpful to be able to explain license conditions using common language and categories that simplify rather than confuse.

## **What Is Open Source Software?**

The simplest way to define open source is that it is software whose source code is freely made available to be used, copied, modified, and distributed. Open source software is offered without requirements for royalty payments. This said, and as emphasized later, open source software can be used for, and often is distributed for, commercial gain, as well as on a not-for-profit basis, depending upon the wishes and/or permissions of its licensors. Contrary to common descriptions, open source software programs also usually are proprietary, in that someone owns them and claims credit for his or her authorship. Indeed, one of the key tenets of the Free and Open Source movement is that credit for authorship should be given where credit is due; almost all open source licenses contain specific instructions on how copyright and attribution notices should be displayed in works that are redistributed.

## **What Is Not Open Source Software Public Domain Software Is Not Open Source Software**

Note that “open source” software is not the same thing as public domain software. Rather, the author or owner of public domain software has abandoned his or her copyright and has contributed material without retaining any right to control how the work is used, reproduced, modified, or distributed. In contrast, open source licensors retain copyright and may impose terms and conditions on the use, copying, modification, and/or distribution of their software.

## **Software That Can Be Downloaded At No Cost Does Not Equal “Open Source Software” or “Free Software”**

Licensees often confuse the term “Free Software” with the concept of “free,” as in no cost for download. Developers frequently download software without having to pay a fee for that download to try it out. At the time of download, there may or may not be a license clearly visible that states whether this “free” code truly is “Free Software” or “open source software”; whether it is licensed on a trial basis or not; whether it has restrictions on its copying, modification, or distribution or not. If such software is not “Free Software” or “open source software” it may be made available with or without kill switches or time bombs that automatically terminate use of the software if the license expires. I have seen situations in which evaluation software was available for no-cost download, but the license contained terms that restricted the licensee from redistributing the code as well as explicit prohibitions on redistributing any code that might have been made available with the download.<sup>9</sup> This “free” (as in no cost for download) software was not “Free Software” nor “Open Source Software” as the FSF or the OSI would have intended those terms. But, if the developer who originally downloaded such software, and then the subsequent developers in the cycle of product development through production, did not read or re-read the applicable license, production versions intended to be distributed on a closed source and/or royalty basis might be issued containing infringing code. In other words, many potential licensees are confused by the availability of software that is available for no cost (that is, “free”) vs. “Free Software.” It is a misconception that, because code is obtained for “free” (at no cost), it always can be modified and/or distributed freely.

## **Let Us Choose Terms That Are Most Useful**

As described already, many terms are used to refer to any and all of “open source,” “free,” “Open Source,” “Free,” “OSS,” “FOSS,” and “FLOSS.” For years I have used the acronym “FOSS” to refer to both “Free” (in the FSF sense) and “Open Source” (in the OSI sense). But the term “FOSS” is less and less in use, and audiences and readers increasingly have questions about what that term means. I have thought that the open source Community should put a decision on one term out to a vote on one of the open source discussion groups, since it would benefit the Community and potential licensees to agree on one term. (To make that happen, however, we would need to agree on *which* discussion list would govern!)

So, for now, I offer these opinions and comments: First, the notion that any software or license that is “open source” needs to have the “Open Source Initiative

# Open Source

---

Approved License” certification attached to it seems exclusionary and exclusive, not open.

Next, when referring to open source software and its associated licenses, we should say “open source” unless, for reasons of historical explanation or precision, we need to distinguish “Free Software” from “Open Source Software.” Software that is “free” (as in “free beer”) may be open source or not and the distinction between what is “free” (as in no cost) and “Free Software” is not always obvious to potential licensees. The term “open source” includes “Open Source” and thus seems more inclusive than “Free.” Van Lindberg has noted that “open source” software is a strict superset of “free software.”<sup>10</sup> Therefore, for purposes of this article and triggering discussion (and except when needed for precise references to certain types of software or licenses), I use here the more inclusive term “open source” rather than the alternatives discussed above. And, in order to contrast open source software and open source licenses from other kinds, I suggest we talk about “closed source” software or “trade secret” source code, with associated “royalty” or “fee”—based licensing.

Assuming that we can all agree on one term to describe open source licensing (which I acknowledge we have not done!), let us look deeper into the well of open source software and licenses and examine some of the buckets used to categorize the software and associated licenses contained there, beginning with the least-neutral terms: “Viral,” “Non-Viral,” “Restrictive,” and “Not Restrictive.”

## **“Viral” vs. “Non-Viral,” and “Restrictive” vs. “Not Restrictive”**

The term “viral” has been used to refer to licenses that require that software redistributed be licensed under the same outbound license conditions as those received inbound. Such license conditions may include making modified or extended versions of the licensed work available in source code form. Companies and licensees that see such release as potentially compromising their businesses tend more to characterize licenses that include requirements of source code availability as “viral” and “non-viral.” Richard Poydner has stated that “[t]his is a subversive condition that some liken to a virus ‘contaminating’ other software code incorporated with it, thereby forcing that code into the commons too.” According to Poydner, Richard Stallman objects to this metaphor: “The GPL’s domain does not spread by proximity or contact, only by deliberate inclusion of GPL-covered code in your program,” he said. “It spreads like a spider plant, not like a virus.”<sup>11</sup> As Heather Meeker points out, the term “viral” is a “pejorative term that is not well received by software developers.”<sup>12</sup> At

this point in the evolution of open source licensing, we need to aim for more neutral terminology.

Similarly, the problem with the words “restrictive” vs. “non-restrictive” is that they are not neutral in describing the nature of licenses. They also have been, and are used to, mean opposite things, depending upon the point of view of the person using them. On the one hand, someone who believes in “Freedom” of software in the FSF sense sees any limitation on such Freedom as “restrictive.” On the other hand, those relying on royalty-based or closed source products to build revenue streams may see “Free” and open source licenses as “restrictive,” particularly when those licenses potentially require the availability of company source code (and related trade secrets) as a condition of distribution. A number of people in the legal and open source communities have sought to distinguish open source software licenses in ways that are non-pejorative to either side, as well as descriptive.

## **“Reciprocal” (“Copyleft” or “Hereditary”) vs. “Academic”**

Some common terms that have evolved to describe licenses such as the GPL include “Reciprocal” and “Copyleft,” as distinguished from the “Academic” or “Permissive” licenses exemplified by the BSD and Apache licenses. Larry Rosen has distinguished open source licenses by distinguishing two basic groups: “Academic” and “Reciprocal” licenses. Rosen explains that “Reciprocal Licenses,” such as the GPL, contain the following bargain: “You may have this free software on condition that any derivative works that you create from it and distribute must be licensed to all under the same license.”<sup>13</sup>

Heather Meeker expands upon this description of “reciprocal”:

The word “reciprocal” is also commonly used to describe free software licenses. Free software licenses are reciprocal in the sense that they require any licensee to “contribute back” to the community any changes the licensee distributes—though this can be misleading, for two reasons. First, this class of licenses may or may not require the source code to be made publicly available; it generally requires the programmer to make modifications available in source code form to all licensees to whom a copy of the binary form of the software is distributed.

Meeker goes on to explain that:

Only a few free software licenses actually require that modifications be contributed back to the original source code tree, and those licenses are

not used frequently. This ambiguity is glossed over because the reciprocity is required in effect; under a free software license, the licensee cannot prevent any of its sublicensees from distributing the software — thus the software is within the reach of the community. However, lawyers use the word “reciprocal” to describe provisions in agreements that place the same obligations on both parties. Free software licenses do not do this at all; most of the conditions apply only to the licensee.<sup>14</sup>

Lindberg, likewise, categorizes open source licenses such as the GPL as “Reciprocal Licenses,” which he describes as the “final stop on the scale of control.” “Reciprocal licenses require that each binary distribution of the code also include full source code to the application.” That is, if licensees incorporate any reciprocally licensed code into an application, they must simultaneously release or make available the full source code for the application. “Additionally, the people releasing code under a reciprocal license must allow others to

## COPYLEFT

Another term that is frequently used to describe the GPL and other reciprocal licenses is “copyleft.” The FSF describes “copyleft” as meaning that “anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has freedom.”<sup>20</sup> Rosen describes the origin of the term “copyleft”:<sup>21</sup>

Partly to emphasize the role of copyright law to protect the freedom of GPL-licensed software and partly to create a catchy term to highlight their focus on software freedom, the authors of the GPL coined the term *copyleft* to describe its license bargain. It is both a play on the word *copyright* and an acknowledgement that it promoted a radical (*i.e.*, left-wing, perhaps) departure from traditional software licensing models. The role of a *copyleft* software license is to grow the public commons of software rather than allow each owner’s copyright to pull from that commons.<sup>22</sup>

The FSF goes on to explain that “[p]roprietary software developers use copyright to take away the users’ freedom; we use copyright to guarantee their freedom. That’s why we reverse the name, changing ‘copyright’ into ‘copyleft.’”<sup>23</sup> But the whole notion of “copyleft” is based upon “*copyright*,” in that one must be a copyright holder or licensee in order to grant the permissions under copyleft licenses. So simply reversing the names does not work. (Aside from the misuse of the term “proprietary,” which is discussed later.)

In short, the term “copyleft” has a cleverness to it that is appealing. In the end, however, the implication that “copyleft” is the opposite of “copyright” is misleading. The term also has taken on a political affect that confuses and distorts what open source licensing is about.<sup>24</sup> To those who are not dealing with open source licensing on a regular basis, this distinction is confusing. For those of us who are introducing open source licensing to others, the need to explain and re-explain this distinction (or lack thereof) is getting all too familiar.

## HEREDITARY

Meeker suggests that we use a new term to replace “reciprocal” and “copyleft” when referring to licenses such as the GPL, LGPL, MPL, and CDDL, noting that that “[t]axonomy is always one of the gating items in writing about open source.” Meeker offers a new term that she hopes “preserve(s) a neutral point of view.”

For a more accurate term, I recommend “hereditary,” because the terms of the license initially applied to the software are “inherited” by all subsequent licensees of the same code, subsets of it, or variations of it. This parallels a concept in object-oriented programming that allows objects or data structures to inherit the characteristics of the parent sets to which they belong.<sup>25</sup>

Meeker defines “hereditary” as a “license with so-called ‘viral’ or ‘copyleft’ terms, sometimes called a ‘free software’ license.” She notes that “this term is not in common usage.”<sup>26</sup> Meeker’s suggestion of the term “hereditary” in lieu of “viral” or “reciprocal” is intriguing, but also suggests a heritage that cannot be shaken and a lack of choice. As described earlier, this enforced heredity seems out of sync with the openness inherent in open source.

freely modify and redistribute their source code under the same reciprocal license. Code is not considered open merely because it is published; it must also be modifiable and distributable.”<sup>15</sup>

But does the notion of “reciprocity” truly require, in the case of software, that “modifications be contributed back to the original source code tree” as Meeker and others state?<sup>16</sup> I do not believe so. “Reciprocity” does not necessarily require giving back exactly the same thing to the same person, but rather making an active choice to “do unto others” in exchange for rights to software. A commonly trusted definition of “reciprocal” is “performed or felt by both sides; mutual” or “complementary.”<sup>17</sup> This choice is in lieu of paying money or undertaking other actively chosen and specified obligations. Of course, and a central point of this article is that, licensees can only make this kind of active choice if they know and understand what they are choosing. In other words, informed consent supports reciprocity; confused consent does not. So we must choose a common language around licensing to help each other understand our choices.

The analogy of code spreading “like a spider plant, not like a virus” is intriguing.<sup>18</sup> Generally, an owner of a spider plant consents to the snipping of one of the offspring and agrees that the recipient of the “snippee” can take home one of those offspring. The original owner wants her own plant and its offspring to thrive, and presumably the recipient of the offspring wants and intends the same.<sup>19</sup> Of course, this notion of “snipping” supports Meeker’s proposed term “hereditary,” but that term is inconsistent with the notion of caregivers of plants/code actively choosing to give and receive offspring—a notion that is more consistent with the idea of openness around open source.

Particularly given the image of gardeners taking “snippets” of mother plants with an implicit promise to foster the offspring, the term “reciprocal” may be preferable when referring to licenses such as the GPL. The notion of reciprocity suggests offering an active choice to each of the licensor and licensee, assuming, of course, that both gardeners know the choices that they are making and intend to foster those under their care.

### **“Academic” and “Permissive” vs. “Reciprocal”**

Larry Rosen and others describe a category of “Academic” licenses, such as the BSD and Apache licenses, distinct from Reciprocal licenses. According to Rosen, these Academic licenses:

promote a . . . kind of freedom, relating to the mission of an academic institution to promote education and scholarship. Teachers are encouraged to

publish their ideas rather than hide them under a cloak of secrecy. Students are expected to take what they learn and apply it to their own work, creating new ideas in turn.<sup>27</sup>

Van Lindberg describes “Academic Licenses” as being the “simplest licenses” that reserve “only the rights of attribution.” (Note that these licenses typically include a disclaimer of warranty as well.) Lindberg cites the MIT and BSD licenses as examples, and notes that the reference to “academic licenses” evolved because such licenses “were originally written for and popularized by universities.”<sup>28</sup> In contrast, Meeker uses the category of “Permissive” to describe “a so-called ‘non-viral’ license, like BSD, MIT, or Apache” and notes that this “term is in relatively common usage.”<sup>29</sup>

---

**None of this means, however, that open source software, by definition, is not or cannot be used in commerce or with “profit as a chief aim.”**

---

While using some of the same categories as Rosen, Meeker and others in the open source Community, Lindberg expands upon these buckets and proposes four groups of licenses: “Academic Licenses,” “Permissive Licenses,” “Partially Closable Licenses,” and “Reciprocal Licenses.” As noted earlier, he describes “Academic Licenses” as being the “simplest licenses” that reserve “only the rights of attribution.” Before defining “Reciprocal Licenses” as the “final stop on the scale of control,” Lindberg modifies the previously-held definition of “Permissive Licenses” as “moving slightly up the scale in control and complexity” in that they “grant substantial rights to downstream users, but include patent, trademark, or public recognition provisions. Works created under these licenses are available for almost all downstream uses, including use in proprietary closed source products.”<sup>30</sup>

Perhaps at this point, and until open source licensing is better understood on a broader scale, we should limit the number of license buckets, and their sometimes conflicting definitions. Perhaps creating a sub-bucket under the category of “Reciprocal,” such as “Fully Reciprocal” and “Partially Closeable” could work. Within the sphere of open source software, I propose that we rely upon the terms “Permissive” and “Reciprocal” when categorizing licenses. At this point, we need to respectfully leave behind the term “Academic,” since many new users of open source software do not understand its historical or scientific reference. The more descriptive term “Permissive” is preferable when

describing licenses such as the BSD license or Apache license, and that term, again, seems more acknowledging of active choice in choosing and/or using a license.

## **The Problem with Current Usage of the Terms “Commercial” and “Proprietary”**

Thus far, I have made suggestions for how we can simplify and clarify terminology used to describe open source. Now, I want to correct a current usage of two terms that is, simply, wrong: “Commercial” and “Proprietary.” These two terms consistently are used in ways that create misunderstanding, and undue distrust, of open source software. The use of these terms in ways that mischaracterize the nature of open source software licensing confuses potential licensees. That confusion gets in the way of adoption of open source software and hinders license compliance.

## **Open Source Can Be and Often Is Commercial**

Frequently, those speaking and writing about open source software refer to “commercial software” and “commercial licenses” in an attempt to distinguish open source software and licenses from software and licenses that are created and marketed for-profit and in business contexts. This usage is just plain wrong.

First, consider the trustworthy definitions of the term “commercial”: The American Heritage Dictionary says that “commercial” means “[of] or pertaining to commerce” and “having profit as a chief aim.”<sup>31</sup> The Oxford English Dictionary says that the adjective “commercial” means “concerned with or engaged in commerce” and “making or intended to make a profit.”<sup>32</sup> There are indeed licenses to freely available software that limit that software’s use for commercial purposes. And there are plenty of not-for-profit open source software projects (the GNU Project, PostgreSQL, Apache Software Foundation, and many others). None of this means, however, that open source software, by definition, is not or cannot be used in commerce or with “profit as a chief aim.” After all, keep in mind that the OSI was formed largely to support the notion that open source software can and should be used for commercial purposes.

Even the FSF is clear on the fact that open source software can be commercial: It states that “commercial software is software being developed by a business which aims to make money from the use of the software.”<sup>33</sup> The FSF further clarifies that

Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software

is no longer unusual; such free commercial software is very important.<sup>34</sup>

Major industry players such as IBM and Sun support open source software, not just as a key part of their business and product strategies but also as contributors to open source “patent commons.” These companies and others such as Oracle, Intel, and Google have been supporting members of open source projects and foundations such as the Eclipse Foundation and the Linux Foundation. The Open Invention Network was formed as an intellectual property company to promote Linux, and that, in its own words, “has considerable industry backing,” including investments from IBM, NEC, Novell, Philips, Red Hat, and Sony.<sup>35</sup>

---

**The term “proprietary” is routinely used in the open source world incorrectly to distinguish closed source software from open source software.**

---

Representatives of many of these and other for-profit companies have had key roles in drafting some of the most prominent open source licenses, including the Mozilla Public License, the Common Public License, and GPL v. 3.0. Other companies, including MontaVista Software, MySQL, SugarCRM, and RedHat have created and built their businesses around the use and distribution of open source software. Still others have formed their revenue streams around open source-related services, such as customization of code stacks or, as in the cases of Black Duck Software and Palamida, assistance with product code review and license compliance. All of these companies are out to make a profit and very definitely are using open source software in a commercial context and for commercial purposes.

In short, open source *can be used for, and frequently IS used for, commercial purposes*. Licensing lawyers, whose goal should be to help clarify license terms for the clients and the public at large, should take the lead by using the term “commercial” precisely when speaking of any kind of software or license. In other words, we should stop using the term “commercial” to mean the opposite of open source.

## **Open Source Software Is Proprietary**

Another term that frequently is used to contrast software made available on a royalty-free, closed source basis from open source software is “proprietary.” The 1971 version of the Oxford English Dictionary defines “proprietary” as “belonging to as proprietor or proprietors; owned or held as property; held in private ownership.”<sup>36</sup>

# Open Source

---

The American Heritage Dictionary defines “proprietary” similarly.<sup>37</sup>

Software that is downloadable, whether in source or object code form, whether for a fee or not—unless it has been abandoned to the public domain—can be “proprietary” in that an individual or some entity owns intellectual property rights associated with that software. That owner may be the author of the software or otherwise have acquired ownership rights in the software through purchase, contribution, or some other transfer. But the term “proprietary” is routinely used in the open source world incorrectly to distinguish closed source software from open source software.

Whereas the FSF gets it right when it comes to clarifying that open source software can be commercial,<sup>38</sup> the FSF adds to the terminology FUD when it portrays proprietary software as the opposite of “Free Software” and open source software:

Proprietary software is software that is not free or semi-free. Its use, redistribution or modification is prohibited, or requires you to ask for permission, or is restricted so much that you effectively can't do it freely . . . we feel there is no possible excuse for installing a proprietary program.<sup>39</sup>

Unfortunately, many authorities on open source software misuse the term “proprietary” as well—sometimes even while acknowledging that the usage is misleading. For example, Lindberg entitles a subsection of one chapter in his book “Proprietary Commercial Licensing” to contrast it with his later discussion of open source licensing.<sup>40</sup> Meeker acknowledges that the term “proprietary” is misleading as it currently is used in open source circles but concludes that she will “bow to convention” when using it. Meeker explains her position:

“Proprietary” describes a license that restricts distribution or use to object code. This term is in nearly ubiquitous usage in the open source community, and so I bow to convention in using it, even though I consider it misleading. Of course, open source is just as proprietary as “proprietary” software, because the software developer's copyright interests are retained. The only truly nonproprietary software, in the copyright sense, is that dedicated to the public domain.<sup>41</sup>

I heartily agree with Meeker that “open source is just as proprietary as ‘proprietary’” software. I respectfully disagree with Meeker, however, that we should perpetuate the use of a term in a way that is inaccurate and misleading. One of the fundamental notions of the

Free and Open Source Software movements has been that credit be given where credit is due and that proper attribution of the author of code be given. All of the prominent open source software licenses require some form of acknowledgement of the copyright holder,<sup>42</sup> and in fact, failure to properly attribute authorship and/or ownership of code is one of the major areas in which licensees frequently trip up.<sup>43</sup> If someone has written or otherwise obtained copyrights to code and has not abandoned it to the public domain, then he or she is the proprietor of that code and that code *IS* proprietary.

## Conclusion: Words Do Matter

In this article, I have offered suggestions for terms that those in the open source community can and should use to clarify, rather than to confuse, the nature of open source software and its licenses. In particular, I hope that readers will choose their words carefully to clarify that both open *and* closed source software may be used for commercial and non-commercial purposes and that any software—other than that contributed to the public domain—is proprietary to its owner.

## Notes

1. In previous articles, I have referred to the various sorts of free and open source software as “FOSS.” As I explain later in this article, I am referring here to “open source software” to cover all such software.
2. GPL stands for the GNU General Public License and LGPL stands for the GNU Lesser General Public License; BSD stands for the Berkeley Software Distribution license, and Apache for the Apache Software License(s). Each of these licenses can be viewed at <http://www.opensource.org/licenses/>.
3. See <http://jnmri.sourceforge.net/k/docket/cafc-pi-1/08-1001.pdf>.
4. A number of examples of attempts to categorize open source licenses can be found at these sites: OSI definition, <http://opensource.org/docs/osd>; OSI license proliferation categorization, <http://opensource.org/proliferation-report>; FSF directory of free software, which includes pictures(!), <http://www.gnu.org/philosophy/categories.html>; FSF categorization of licenses, <http://www.fsf.org/licensing/essays/categories.html>; Sun's Categories A, B, and C, [http://www.sun.com/software/opensource/whitepapers/Sun\\_Microsystems\\_OpenSource\\_Licensing.pdf](http://www.sun.com/software/opensource/whitepapers/Sun_Microsystems_OpenSource_Licensing.pdf).
5. See <http://sourceforge.net/>. See also <http://alexandria.wiki.sourceforge.net/What+is+SourceForge.net%3F>.
6. See <http://www.opensource.org/licenses/>.
7. See <http://www.linux-watch.com/news/NS5369154346.html>; [http://news.cnet.com/8301-13505\\_3-9910263-16.html](http://news.cnet.com/8301-13505_3-9910263-16.html); <http://marketshare.hitslink.com/report.aspx?qprid=8>; and <http://marketshare.hitslink.com/report.aspx?qprid=0>.
8. See Murray, Gwyn Firth, “Getting with the Program: A Guide for Lawyers Working with Free and Open Source Software

- in the Enterprise” (Oct. 2005), at <http://www.mataulegal.com/publications.php>.
9. See as an example the Java™ Media Framework (JMF) 2.1.1b Binary Code License Agreement at <http://repository.jboss.org/licenses/sun-jmf.txt>.
  10. See Lindberg, Van, *Intellectual Property and Open Source: A Practical Guide for Protecting Code* (O’Reilly 2008).
  11. Richard Poydner, “Reclaiming the Digital Commons,” quoting Richard Stallman, at [http://www.richardpoynder.co.uk/reclaiming\\_the\\_digital\\_commons.htm](http://www.richardpoynder.co.uk/reclaiming_the_digital_commons.htm). Many thanks to Richard Fontana for bringing Stallman’s analogy to my attention; at the time of this writing, I had not yet found the original quote.
  12. Meeker, Heather J., *The Open Source Alternative: Understanding Risks and Leveraging Opportunities* (John Wiley & Sons, 2008) at p.11.
  13. Rosen, Lawrence, *Open Source Licensing: Software Freedom and Intellectual Property Law* (Prentice Hall PTR 2005), p.103.
  14. Meeker, Heather J., *The Open Source Alternative: Understanding Risks and Leveraging Opportunities* (John Wiley & Sons, 2008) at pp.11-13.
  15. Lindberg, Van, *Intellectual Property and Open Source: A Practical Guide for Protecting Code* (O’Reilly 2008) p.203.
  16. Ref. Fontana, Richard; Rosen, Larry; Bain, Malcolm, and others in active discussion, at time of submission of this article for publication on fsfeurope.org [FTF-Legal] email mailing list.
  17. See the definition of “reciprocal” in the American Heritage Dictionary, 4th edition, (Houghton Mifflin 2001). This dictionary also offers this definition of the word “reciprocate: to show or give in return.”
  18. Richard Stallman, as quoted by Sam Williams: “To compare something to a virus is very harsh,” says Stallman. “A spider plant is a more accurate comparison; it goes to another place if you actively take a cutting.” See Williams, Sam, *Free as in Freedom: Richard Stallman’s Crusade for Free Software* (O’Reilly 2002) Chapter 2, n.1. See <http://oreilly.com/openbook/freedom/ch02.html>. Richard currently is Open Source Licensing, Copyright and Patent Counsel to Red Hat, Inc., and former Counsel to the Software Freedom Law Center.
  19. The snipping does not change the original spider plant, nor does it change the nature of the offspring, except that the offspring now has a chance for independent life and growth. The person owning the mother spider plant, one hopes, continues to care for it and foster its well being. The person snipping the offspring, one hopes, will care for it and foster its growth. The offspring, if the snipper waters and plants it well grows to be its own plant. If the snipper adds food dye to either plant’s water, the plant may change color. (I know this works with carnations.) If the baby spider plants grow and send out new baby spider plant shoots, those remain spider plants. But being snipped in and of itself does not turn a spider plant into another plant, nor does the baby spider plant’s own growth change the nature of other plants around it. If the owner plants a baby spider plant in the same planter with a philodendron, the spider plant and the philodendron retain their original properties. If the “snipper” gives away or sells its now-independent and bigger spider plant, that plant remains a spider plant. And, if the owner gives away or sells the now-independent offspring in the same pot with a philodendron, both the philodendron and the spider plant retain their own basic identities.
  20. See <http://www.fsf.org/licensing/essays/copyleft.html>.
  21. The FSF’s description of “copyleft” is that “anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it.” See <http://www.fsf.org/licensing/essays/copyleft.html>.
  22. Rosen, Lawrence, *Open Source Licensing: Software Freedom and Intellectual Property Law* (Prentice Hall PTR 2005), p.105.
  23. See <http://www.fsf.org/licensing/essays/copyleft.html>.
  24. Ref FTF-Legal mailing list exchanges on 8/31/2008, including discussion raised by Richard Fontana of whether the term “copyleft” itself “carries much ideological content today.” I believe it does, and this is part of why I prefer other terminology.
  25. Meeker, Heather J., *The Open Source Alternative: Understanding Risks and Leveraging Opportunities* (John Wiley & Sons, 2008) at pp.11-13.
  26. *Id.*
  27. Rosen, Lawrence, *Open Source Licensing: Software Freedom and Intellectual Property Law* (Prentice Hall PTR 2005), p.73.
  28. Lindberg, Van, *Intellectual Property and Open Source: A Practical Guide for Protecting Code* (O’Reilly 2008) p.201.
  29. Meeker, Heather J., *The Open Source Alternative: Understanding Risks and Leveraging Opportunities* (John Wiley & Sons, 2008), Preface at X.
  30. Lindberg, Van, *Intellectual Property and Open Source: A Practical Guide for Protecting Code* (O’Reilly 2008) p.201. Lindberg goes on to define a third category of open source licenses, the “Partially Closable Licenses,” which “allow the creation of proprietary, closed source systems using the code while still mandating some code sharing.” He cites the MPL (Mozilla Public License) and the LGPL as being these kinds of licenses. Lindberg explains that, in contrast to the “Academic” or “Permissive” licenses, which permit the licensee to re-use and redistribute the original or modified code under either an open source or a closed source license, these “Partially Closable Licenses” divide the covered code into two pieces: “All code associated with the first piece must remain open source. The second piece can be distributed without sharing the source code. Because there is a required separation between the open and closable parts of the application, partially closable licenses usually show up in two areas: libraries and extensible applications.” (At pp.201-202). I think that this is an interesting notion, and I have referred to such licenses for my own purposes and to clients as “modular” licenses. But, as explained here, for now I suggest we reduce, rather than expand the number of license “buckets” at least until the central tenets of open source licensing are understood clearly by a broader audience.
  31. This dictionary’s more complete definition is “1. Of or engaged in commerce 2. Having profit as a chief aim. 3. Supported

# Open Source

---

- by advertising,” (American Heritage Dictionary, 4th edition, Houghton Mifflin 2001).
32. See [http://www.askoxford.com/consider\\_oed/commercial?view=uk](http://www.askoxford.com/consider_oed/commercial?view=uk). Also see The Compact Edition of the Oxford English Dictionary, which defines “commercial” as “engaged in commerce; trading. Having reference to or bearing on commerce. . . viewed as a mere matter of business; looking toward financial profit.” (Oxford University Press, 1981 (20th printing) p.480.)
  33. See <http://www.fsf.org/licensing/essays/categories.html#commercialSoftware>.
  34. See <http://www.gnu.org/philosophy/free-sw.html>.
  35. See <http://www.openinventionnetwork.com/about.php>.
  36. “. . .B. adj. 1. Belonging to a proprietor or proprietors; owned or held as property; held in private ownership. . .” The Compact Edition of the Oxford English Dictionary (Oxford University Press 1971) at p.1483.
  37. “1. Of or pertaining to a proprietor or proprietors collectively. 2. Exclusively owned; private. 3. Befitting an owner [or] 4. Owned by a private individual or corporation under a trademark or patent. . .” The American Heritage Dictionary, Second College Edition (Boston: Houghton Mifflin Company 1982, 1985).
  38. See the FSF’s statement that “[c]ommercial’ and ‘proprietary’ are not the same thing! Most commercial software is proprietary, but there is commercial free software, and there is non-commercial non-free software.” See <http://www.fsf.org/licensing/essays/categories.html>.
  39. See <http://www.fsf.org/licensing/essays/categories.html>.
  40. Lindberg, Van, Intellectual Property and Open Source: A Practical Guide for Protecting Code (O’Reilly 2008) p.200. Mr. Lindberg further explains that, under “proprietary” licensing schemes “the person receiving the software is usually granted only the rights to install a single copy and run the software on a single computer. In these licenses, there are so few rights granted to downstream users that it is relatively easy for the license to specify the few allowed uses . . . More complex are time-based, seat-based, or other “flexible” proprietary licensing schemes. . . charging different users different licensing fees according to their specific needs and ability to pay. . . There are also some proprietary licenses that allow redistribution of some closed source components . . .” p.200.
  41. Meeker, Heather J., The Open Source Alternative: Understanding Risks and Leveraging Opportunities (John Wiley & Sons, 2008) at pp.11-13.
  42. See the license attribution terms in licenses listed at <http://www.opensource.org/licenses>.
  43. For more detail on this subject, see my paper, “Attribution Requirements: Free and Open Source Software Licensing” (Sept, 2007) at [www.mataulegal.com/publications.php](http://www.mataulegal.com/publications.php).

Reprinted from *The Computer & Internet Lawyer*, January 2009, Volume 26, Number 1, pages 1 to 11, with permission from Aspen Publishers, Inc., a Wolters Kluwer business, New York, NY, 1-800-638-8437, [www.aspenpublishers.com](http://www.aspenpublishers.com).