

California Business Law PRACTITIONER



Gwyn Firth Murray 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 technology industries (see <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 <http://www.open-bar.org>). She is a graduate of Stanford University Law School and also holds an M.A. in Latin American Studies from Stanford University. She obtained her B.A. magna cum laude and with distinction in economics from Yale College. Ms. Murray is fluent in Spanish, English, and Portuguese.

Open Source: A Challenge Worth Meeting

Gwyn Firth Murray

Introduction

Since the author first started working with open source software, there has been a staggering increase in the amount of software that is freely available and in the number of associated licenses. A decade ago, the world of open source licensing was limited to only a few licenses. The most widely used and recognized was the governing license of Linux, the GNU General Public License (GPL), along with its “lesser” license, the GNU Lesser General Public License (LGPL). Other important and commonly used licenses were the Berkeley Software Distribution (BSD) license and the Apache license. (Each of these licenses can be viewed at <http://www.opensource.org/licenses/>.)

Since then, new open source applications and corresponding licenses have multiplied exponentially, and the task of understanding the terms that govern the wide variety of open source licenses has become more of a challenge. It has also become

—continued on page 107

IN THIS ISSUE

Open Source: A Challenge Worth Meeting	105
by Gwyn Firth Murray	
Contractual Arbitration in California	113
by George W. Kunej	
Insurance “Bad Faith” Basics, Part II	123
by Guy O. Kornblum	
Buying or Selling a Winery, With Annotated Letter of Intent	130
by Richard Lemon	

CEB

CONTINUING EDUCATION OF THE BAR □ CALIFORNIA

© 2009 by The Regents of the University of California

critical from a licensee's perspective to understand and comply with open source license terms. Recent court actions, including last year's Federal Circuit decision in *Jacobson v Katzer* (Fed Cir 2008) 535 F3d 1373, have supported the enforceability of open source licenses. Increasingly, organizations such as the Software Freedom Law Center (see <http://www.softwarefreedom.org>) are taking formal action on behalf of open source licensors—with reported success in terms of settlements conditioned on the defendants' publishing applicable source code and payment of financial consideration.

Given the increased use and popularity of open source software, companies are now more aware of the issues involved in licensing open source code, and more potential customers, acquirors, and investors are requiring stringent guarantees of licensees' compliance with license terms. Thus, even without the threat of increased litigation, it is more important than ever that companies using open source software clearly understand the terms on which they bring that software into their operations and how to comply with those terms. This article is intended to provide a brief introduction to open source software licensing and to offer some tips to practitioners on how to promote license compliance by their client companies.

WHAT IS OPEN SOURCE SOFTWARE?

The simplest way to define open source software is that it is software, the source code for which is made freely available to be used, copied, modified, and distributed. Open source software is offered without requirements for royalty payments. Nonetheless, open source software can be used, and often is distributed, for commercial gain—as well as on a not-for-profit basis—depending on the wishes of (or permissions granted by) its licensors. Open source software programs also are proprietary in that someone owns the copyrights to them and claims credit for authorship. Indeed, one of the key tenets of the open source movement is that credit for authorship should be given where credit is due.

A BRIEF HISTORY OF OPEN SOURCE SOFTWARE

It is helpful to remember that neither “free” software nor open source software is new to computer technology.

NOTE: In this article, the word “free” in quotation marks is used to refer to software that the li-

censee has the freedom to run, modify, and distribute; it does not refer to software that is not “free” in this sense and that is merely being distributed without cost.

More than 30 years ago, software development was considered part of the scientific enterprise and was developed with much input and sharing among developers, mostly within academic, government, and other research institutions. The operating system UNIX, designed to run large mainframe computers, was developed at AT&T's Bell Labs, along with C, the programming language used to write it. At first, UNIX was licensed by Bell Labs on a noncommercial basis. But then UNIX was “forked” and versions of it were licensed on a closed source basis by a group of companies that included IBM and Sun Microsystems.

In the meantime, Apple Computer and IBM were developing their personal computers, each of which required a less cumbersome operating system than UNIX. Hence, Apple developed its Macintosh operating system, Microsoft developed what is now Microsoft Windows for IBM, and IBM, Apple, Microsoft, and other companies discovered that they could charge royalties for these closed source software operating systems and applications. These developments marked the emergence of what is now considered traditional software licensing, which is also often (and not always correctly) referred to as proprietary or commercial software licensing.

Meanwhile, Richard Stallman and other computer scientists were displeased by the closing of UNIX source and the growth of royalty-based software development and licensing. Stallman founded the GNU Project (see <http://www.gnu.org/>) in order to build an alternative to UNIX, and he wrote the original GNU GPL. Linus Torvalds wrote the Linux operating system and then agreed that Linux could become part of the GNU Project, which was placed within the bailiwick of Stallman's Free Software Foundation (FSF) (see <http://www.fsf.org>). The FSF is the original author and keeper of the GPL (currently in version 3.0, see <http://www.gnu.org/licenses/gpl-3.0.txt>) and remains the key proponent of the notion of “free” software.

According to the FSF, the term “free” software means that the licensee has the freedom to run, modify, and distribute the software. However, “free” does not necessarily mean without cost, in that fees may be charged to cover costs and services related to the distribution of “free” software. To understand the notion of “free” software, one must understand that the freedom of the software runs with the pro-

gram, not with the licensee. As the FSF has explained, one should think “free speech, not free beer.” See <http://www.gnu.org/philosophy/free-sw.html>.

Open source software, a group of software programs that includes “free” software, 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 on 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 they were talking about: “open source.” The “free” and open source software movements share common goals, in that they both focus on the free redistribution of programs and the requirement to make source code available. It is important to understand, however, that these movements grew from different perspectives. Open source software generally is seen as more commercial-leaning and inclusive than “free” software.

“FREE” SOFTWARE VERSUS OPEN SOURCE SOFTWARE

The “free” software movement is either more encouraging, or more restrictive, of licensees’ freedom than the open source movement—all depending on one’s perspective. Under FSF guidelines regarding what constitutes “free” software, all distributed modifications to the licensed software must be licensed on the same terms under which the software was originally received. This kind of requirement has been referred to as “copyleft,” “viral,” “reciprocal,” or “hereditary,” and is what understandably strikes the most fear in the hearts of some commercial businesses. If viral open source software enters a company without proper attention to the licensing terms that govern it, a company’s entire source code base—or that of a key product line—could be “contaminated” and required to be made available to the general public.

NOTE: The FSF defines “copyleft” to mean that “anyone who redistributes software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has that freedom.” See <http://www.fsf.org/licensing/essays/copyleft.html>. Contrary to the common misconception that “copyleft” is the opposite of “copyright,” the whole notion of copyleft is based on copyright, in that one must be a copyright holder or licensee in order to grant the permissions under copyleft licenses.

Open source software is similar to “free” software in that its licensees are entitled to access, use, copy, modify, and distribute open source and binary code without making royalty payments to the licensor and to combine open source software with other software code. But software that is defined as open source may be licensed with fewer restrictions (or freedoms, depending on one’s perspective) on its downstream distribution. Open source licenses may be reciprocal or copyleft licenses, yet may permit the licensee to take the licensee’s code modifications private or closed source. These latter types of licenses are now being referred to as “academic,” “non-viral,” or “permissive.”

The most common and recognized definition of open source is the Open Source Definition (OSD) created and maintained by the Open Source Initiative (OSI) (see <http://www.opensource.org/>). The OSD clarifies that “[o]pen source doesn’t just mean access to the source code” and requires that a license must comply with ten specified criteria in order to qualify as an OSI-approved license. See <http://www.opensource.org/docs/osd>. Although 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 make software 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.

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. The author or owner of public domain software has abandoned his or her copyright and has contributed material to the public domain without retaining any right to control how the work is used, reproduced, modified, or distributed. In contrast, open source licensors retain their copyright and may impose terms and conditions on the use, copying, modification, or distribution of their software.

Software That Can Be Downloaded for Free Is Not Necessarily Open Source or “Free”

In the author’s experience, licensees often confuse the term “free” with the concept of being free of cost, with sometimes unfortunate results. Developers may 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” or open source, whether it is licensed on a trial basis, or whether it has restrictions on its copying, modification, or distribution. If not “free” or open source, it may be made available with or without “kill switches” or “time bombs” that automatically terminate use of the software if the license expires. The author has seen situations where evaluation software was available for no-cost download, but the license contained terms that restricted the licensee from redistributing the code, including explicit prohibitions on redistributing any code that might have been made available with the download. See, e.g., the Java™ Media Framework (JMF) 2.1.1b Binary Code License Agreement at <http://repository.jboss.org/licenses/sun-jmf.txt>. This free software was not “free” or open source as the FSF or the OSI would have intended the meaning of those terms. But if the developer who originally downloaded the software and the subsequent developers in the cycle of product development did not read or re-read the applicable license, production versions intended to be closed source might be issued containing infringing code. The misconception that if code is obtained at no cost it can be modified

or distributed freely is a compliance pitfall for unsuspecting companies.

THE PATH TO COMPLIANCE

Numerous papers discuss best practices concerning open source software. See, e.g., Murray, *Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise* (October 2005) 3–4, available at <http://www.mataulegal.com/articles.htm>. In this paper, the author offers these general tips:

Remember That “Software Is Software”

Companies using open source software need to recognize that very similar rules of due diligence and license compliance apply whether they are dealing with closed source or open source software. Even if all inbound code is subject to a closed source commercial license, licensee companies should have policies and procedures in place for managing the intake of that software and the associated intellectual property rights. In addition, they should follow up and maintain those policies. When considering whether to use third party code, a company should be thinking about how that code fits within its overall business plan and strategy, its future product lines, and its distribution and revenue models. With those considerations in mind, the company should carefully select the vendor products required or useful to implement its particular product development and marketing strategy. The company should perform a technical review and evaluation of potential incoming third party code for suitability within its own product lines and should review or negotiate applicable licenses to ensure that it has the rights it needs throughout the evaluation, development, production, and distribution processes. Finally, every company should have systems in place to monitor and enforce compliance with the license terms and to track software acquisition and development during all phases of its own technology development and production.

Check the Source of the Source

As part of the analysis described above, companies seeking to manage their intellectual property assets in the age of open source software should ensure that the technical and business review they conduct with respect to incoming code covers such questions as: What is the “pedigree” of the code? Who wrote it, and can the company confirm its ori-

gin? Does it contain any third party code? If so, has the origin of that code been established? Have all necessary third party rights been transferred? And what license terms apply to that third party code?

Companies should ask these questions no matter where their inbound code comes from. There is an important difference, however, between using closed source and open source code, particularly when using open source code governed by reciprocal or viral license provisions such as those of the GPL. If a company mingles its closed source code with GPL'd code and then distributes that code, it may trigger requirements to make company and other proprietary source code publicly available. It also risks both (1) the unintended loss of critical company intellectual property and (2) infringement claims from open source or other third party software providers. One of the worst nightmares surrounding open source software is the story of Linksys, which Cisco Systems purchased in 2003 for \$500 million. Three months after the purchase, complaints appeared on "free" and open source software (FOSS) discussion boards claiming that Linksys had violated a key FOSS license by not releasing source code found in its product. Several months and a lot of negotiation later, Linksys publicly released the subject source code. One lesson from the Linksys story is that companies need to be alert to the possibility that they have open source software in their code bases already, like it or not, and that they may already have unwittingly donated some of their intellectual property to the broader community. Had Linksys and Cisco thoroughly checked the source of the source upstream, they might have avoided the nightmare of releasing valuable source code. The Linksys case points out

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

Meeker, *The Legend of Linksys* (LinuxInsider, January 1, 2006), available at <http://www.linuxinsider.com/story/47987.html>.

The Linksys example highlights the importance of companies' establishing and diligently maintaining processes for monitoring third party license terms and complying with those terms throughout the software development process, including the evalua-

tion, testing, development, production, and distribution stages. In particular, companies need to ensure that they have adequate and necessary rights for any downstream licensing and redistribution. They should also ensure that they have confirmed any necessary rights to offer technology on an application service provider (ASP) or other basis. Finally, companies need to be alert to open source software licensing requirements, particularly the reciprocal provisions of any licenses governing incoming code, throughout all stages of software development and distribution.

Recognize That Working With Open Source Software Requires Cultural Change

Working effectively with open source software requires cultural change—among companies, their employees, and their lawyers. There are vast cultural and practical differences between working with and within the open source software community and working with closed source software. Chances of implementing an effective company policy are increased when lawyers are willing to learn about open source software, to get "down and dirty" by understanding technical specifications and software functionality, and to adapt to the particular cultures of the client company and the open source software community.

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

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

Wacha, *Open Source, Free Software, and the General Public License*, 10 *The Computer Lawyer* 3 (Mar. 2003).

Although closed source licenses have companies and their lawyers behind them, they generally do not have a passionate community around them to help support enforcement efforts. In contrast, the open source software community sees as part of its mission the interpretation and enforcement of open source software licenses. This interpretation and enforcement takes place through often very blunt dialogue on public websites. Therefore, building and maintaining relationships with open source software developers and the open source software community can be helpful. If a company is known to the open source software community as a responsible licensee that endeavors to comply with open source software license terms, the “flames” posted on open source software-related websites may be kinder and gentler.

Understand That Licensing Isn't Just for Lawyers Anymore

Part of the cultural change associated with the prevalence of open source software means that lawyers need to be connected to the open source community and to their developer clients. Lawyers used to be company gatekeepers without whose approval no software could come in the door. Now, however, engineers can, and do, surf the Web and freely download to their computers at work and at home a wide variety of software, which comes in as both source and object code and which is governed by a potentially huge variety of license terms. When software comes in as source code, developers' ability to modify and create derivative works from it increases, which means it is easier to commingle with proprietary code, including a company's own code and other third party code. The consequences of such commingling—particularly with respect to open source software code governed by copyleft license provisions such as those in the GPL—can be severe. For example, a requirement to make company and other proprietary source code publicly available on distribution of commingled code may be triggered.

To be successful in this new world of open source software, lawyers for commercial businesses must recognize that there is no more ivory tower. Many software developers know—and even more *think* they know—much more about open source software and licensing than the average high tech company lawyer. Until recently, software engineers have enjoyed a lot of freedom to download, use, and play

with open source software without questions or interference from lawyers. They have also been writing licenses, sometimes without any input from the legal profession, and sometimes those licenses include technical restrictions that the average lawyer will not understand at first glance. Many developers have been working with open source software a lot longer than lawyers have.

Understandably then, developers may resist lawyers' efforts to intervene and create policies that restrict their freedom to play with the exciting variety of software publicly available or to release software that they develop on the terms they choose. This places a burden on lawyers to build relationships with the engineering community (and particularly the open source software community) as never before. Given the vast array of software available and the sometimes bewildering license terms under which it is made available, lawyers wanting to learn about open source software will need to ask for help from developers in order to learn about available open source software technology and to interpret the associated variety of sometimes quite technical licensing terms. Being an effective lawyer in the world of open source software requires a high degree of technical knowledge. Lawyers should therefore start asking questions and asking for help. If an attorney demonstrates that he or she is genuinely interested in the technology being brought into and developed at the company and is open to the company's use of open source software and engagement of the open source software community, the attorney will likely find the engineers more receptive to his or her involvement.

Read — and Re-Read — Each and Every License

Reading the license may sound so obvious that it does not need to be stated, but in the author's experience, not reading the license carefully is where many licensees trip up. Closed source licenses are also subject to a multitude of different terms, but those terms are usually subject to nondisclosure requirements and are not publicly available for everyone to see. Further, the terms have usually been written by other lawyers in more conventional legal language. However, just as a licensee carefully reviews closed source software licenses before accepting them, licensees of open source software must review applicable license terms in light of the licensee's intended use of the software. If it is not clear what the terms mean or what they permit, the lawyer should find a friendly software developer or IT pro-

fessional and ask for help understanding how the software works, how it links, how it compiles, and how it is distributed from a technical perspective.

Many companies think the GPL is their worst nightmare because it is both widely used and has strong copyleft attributes. In the author's experience, however, the GPL is not the open source software lawyer's greatest problem. Most software developers now understand the copyleft implications of the GPL and how to navigate around them effectively. In fact, the author finds that more companies get into trouble with respect to other licenses, whether open source or closed source. The main reason they do is because they have not had a process in place (1) for keeping proper records of (a) which license applies, *i.e.*, what license was in effect on the date the software was downloaded, and (b) where that license is, *i.e.*, whether a copy is still available on the Web and whether the company has a hardcopy stored, and (2) for making sure that the permissions under the license continue to match the company's needs throughout the development, production, and distribution process. Many engineers understand the implications of downloading open source software and have read—and care enough to abide by—the terms of their employment agreements and the applicable license before they download. However, even the best-informed and best-intentioned developer may forget about license terms that, *e.g.*, permitted testing and evaluation but prohibited modification or distribution, by the time the developer starts incorporating the third party code and using it to develop product. Here is where a well-thought-out intellectual property management program will help keep the company out of trouble.

Give Credit 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. At a minimum, most FOSS licenses include a requirement that a licensee redistributing the licensed code (or a modified version of the code) provide a basic copyright (or copyleft) notice acknowledging the identity of the original author(s). Attribution and notice requirements, however, vary widely among licenses. Many licenses (*e.g.*, the GPL and the BSD license) also include specific requirements concern-

ing how notice of the license is to be given. Some open source licenses include a requirement that each contributor-licensee of the code provide notice of modifications made by that contributor to each version of the licensed software. Licenses stating such a requirement include the GPL, the LGPL, the Apache license, the Mozilla Public License (MPL), the Common Development and Distribution License (CDDL), the Perl license, the Academic Free License (AFL), and the Open Software License (OSL). In contrast, the Common Public License (CPL), the BSD license, the Massachusetts Institute of Technology (MIT) license, and the Eclipse Public License (EPL) do not include a requirement that licensees who modify code provide notice of their changes. Paying attention to the copyright and license notice requirements of the licenses governing the client company's code reassures the open source community that the company is endeavoring to comply with those licenses, and buys the company goodwill within the community.

CONCLUSION

There are valid reasons why fear, uncertainty, and doubt linger around the use of open source software in commercial enterprises. Working with open source software requires that lawyers acquire more technical knowledge and be more diligent about implementing and maintaining best practices programs than in the past. It also means that lawyers must build relationships with clients and other lawyers as never before. If attorneys are to be effective, they need to be open to receiving help from developers who may have more in-depth knowledge of open source software technology and licensing terms. In a world where almost no statutory or case law on open source software exists, lawyers need to acknowledge and accept the power of the open source community. If companies are alert to the issues presented by bringing open source software into their enterprise and manage its implementation thoughtfully, they can avoid unnecessary risks. Given the potential that open source software offers for developing innovative technology “more, better, faster,” companies are likely to find that opening up their enterprise to open source software is a sound business strategy.