Intellectual Property Litigation

FROM THE SECTION OF LITIGATION INTELLECTUAL PROPERTY LITIGATION COMMITTEE

Fall 2011, Vol. 23 No. 1

TABLE OF CONTENTS

Articles »

Intellectual Property Litigation
FROM THE SECTION OF LITIGATION INTELLECTUAL PROPERTY LITIGATION COMMITTEE

Fall 2011, Vol. 23 No. 1

ARTICLES

Obtaining Preliminary Injunctions in Open-Source Cases

By S. Christian Platt, Bob B. Chen, and Kenneth Newton – September 20, 2011

On August 13, 2008, the Federal Circuit held in *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008) (*Jacobsen I*), that parties who distribute software under the terms of an open-source license may obtain copyright remedies, including a preliminary injunction, against defendants who breach the terms and conditions of their license. On January 5, 2009, on remand, the U.S. District Court for the Northern District of California denied for a second time the plaintiff's motion for a preliminary injunction to stop a defendant from violating the terms of an open-source license. *Jacobsen v. Katzer*, 609 F. Supp. 2d 925 (N.D. Cal. 2009) (*Jacobsen II*). In so doing, the district court rejected the prevailing rule in copyright cases that entitled plaintiffs to an automatic presumption of irreparable harm upon showing a reasonable likelihood of success on the merits.

The *Jacobsen II* court's interpretation of a higher standard of proof and the move by other courts away from the presumption of irreparable harm casts doubt on whether open-source plaintiffs can ever practically meet the preliminary injunction standard. Although economic considerations may exist in the open-source licensing context beyond that of traditional license royalties or lost profits, such damages are more difficult to prove. In practice and given the lack of available evidence, the further open-source plaintiffs deviate from a profit-based software model, the harder they will find it to obtain a preliminary injunction.

Obtaining a Preliminary Injunction and Proving Irreparable Harm

The Copyright Act, 17 U.S.C. § 502(a) (2011), authorizes courts to issue an injunction to "prevent or restrain infringement of a copyright." A plaintiff must satisfy a four-factor test to obtain a preliminary injunction under the Copyright Act. First, the plaintiff must show that he or she is likely to prevail on the merits. Second, the plaintiff must show that he or she is reasonably likely to suffer irreparable harm absent an injunction. Third, the balance of equities must favor the plaintiff. Finally, imposing the injunction must be in the public interest.

In general, an irreparable harm is defined as an injury that the plaintiff suffers for which the court cannot compensate should the plaintiff prevail in a final decree. Economic losses that are calculable and compensable by monetary damages ordinarily do not constitute an irreparable injury. In contrast, courts have found irreparable harm in those circumstances where losses have been difficult to calculate or impossible to measure—for instance, where a plaintiff has suffered damage to his or her reputation or where the calculation of lost sales is too speculative.

At the time of the Federal Circuit's *Jacobsen I* decision the prevailing view among courts, including the Ninth Circuit, gave plaintiffs the benefit of a presumption of irreparable harm in cases where the plaintiff was able to show a reasonable likelihood of success on the merits of a claim of copyright infringement. Pursuant to this approach, so long as they could show that they were likely to succeed on their infringement claims, plaintiffs were not required to offer a detailed showing that they would be irreparably harmed. 4 Melville B. Nimmer and David Nimmer, *Nimmer on Copyright* § 14.06[b] (Matthew Bender, Rev. Ed.)

The Federal Circuit Upholds the Enforceability of Open-Source Licenses
In *Jacobsen I*, 535 F.3d at 1382–83, the Federal Circuit held that open-source licenses are enforceable under copyright law. The plaintiff, Jacobsen, managed an open-source software group that developed DecoderPro, an application to program decoder chips that controlled model trains. Under the terms of the artistic license, one of many available open-source licenses, the group released the application for public use. The defendant, Katzer, developed and patented a proprietary competing application that allegedly incorporated portions of the DecoderPro program. The software was released without complying with the terms of the artistic license. Specifically, the software files Katzer used that incorporated the DecoderPro program did not provide proper attribution to Jacobsen's code, did not reference how that code had been changed, did not include the authors' names, did not include Jacobsen's copyright notices, and did not include a reference to a file that contained the terms of the artistic license.

Jacobsen brought suit seeking a preliminary injunction against Katzer for breach of contract and copyright infringement and seeking a declaratory judgment that Katzer's patent was invalid. At the center of the dispute was whether the key limitations set forth in the artistic license created covenants governed by contract law, conditions governed by copyright law, or both. If the limitations set forth created only covenants, Jacobsen would generally be limited to monetary damages. But if the limitations set forth conditions, Jacobsen could recover remedies under copyright law, including a preliminary injunction.

In the underlying district court proceeding, the Northern District of California held that while the terms of the license had been breached, the terms created covenants under contract law and not conditions under copyright law. In addition, Jacobsen was not entitled to monetary damages, as he had distributed the DecoderPro software for free. *Jacobsen v. Katzer*, No. C 06–01905 JSW, 2007 WL 2358628, at *6–7 (N.D. Cal. 2007). The Federal Circuit heard the case on appeal under its supplemental jurisdiction in light of Jacobsen's related patent claim. In deciding whether to grant Jacobsen's motion for a preliminary injunction, the Federal Circuit interpreted Ninth Circuit law to determine the copyright and contract issues.

The Federal Circuit reversed the decision of the district court, interpreting the terms of the license as including conditions giving open-source developers copyright protection for their software. It held that copyright owners "who engage in open source licensing have the right to control the modification and distribution of copyrighted material" and that, absent the right to obtain a preliminary injunction, open-source license restrictions "might well be rendered meaningless." *Jacobsen I*, 535 F.3d at 1381–82. Under Ninth Circuit law, Jacobsen was further entitled, upon showing a reasonable likelihood of success on the merits, to a presumption of irreparable harm.

The Federal Circuit further held that the lack of monetary exchange in the issuance and use of an open-source license should not be presumed to mean that there are no substantial economic benefits to the creation and distribution of copyrighted works under public license. For example, software developers could save the time and costs of improving software by making components available to the public free of charge. Releasing software under an open-source license would also grant developers the ability to generate market share and build a software developer's reputation. *Id.* at 1379.

The Federal Circuit remanded the case to the Northern District of California to decide whether Jacobsen's preliminary injunction motion was consistent with its determinations.

The Effect of the Heightened Standard of Proof

On remand, the Northern District of California denied Jacobsen's request for a preliminary injunction in *Jacobsen II*, 609 F. Supp. 2d 925. The district court held that, while open-source plaintiffs could be granted a preliminary injunction, the Supreme Court's intervening decision in *Winter v. National Resources Defense Council, Inc.*, 555 U.S. 7, 129 S. Ct. 365, 374 (2008), had created a heightened standard of proof requiring the plaintiff to demonstrate the injunction's necessity. In *Winter*, the Supreme Court emphasized that a preliminary injunction is an extraordinary remedy that "may only be awarded upon a clear showing that the plaintiff is entitled to such relief" and that a plaintiff must show that irreparable injury is "likely in the absence of an injunction." *Id.* at 367, 376.

Relying on the *Winter* decision, the district court held that plaintiffs were no longer entitled to a presumption of irreparable harm upon showing a reasonable likelihood of success on the merits of their copyright infringement claims. *Jacobsen II*, 609 F. Supp. 2d at 936. Instead, the district court held that the plaintiff must meet all four factors of the preliminary injunction test. The *Jacobsen II* court then denied Jacobsen's motion for a preliminary injunction a second time, as Jacobsen had proffered no admissible evidence that he had suffered irreparable harm; he only demonstrated that such harm was possible.

Other recent court decisions have also cast doubt on whether the presumption of irreparable harm still applies in copyright cases. On April 30, 2010, in *Salinger v. Colting*, 607 F.3d 68 (2d Cir. 2010), the Second Circuit provided a more thorough analysis for abandoning the presumption of irreparable harm for preliminary injunctions in copyright cases. Relying on the Supreme Court decision *eBay v. MercExchange, LLC*, 547 U.S. 388, 392 (2006), which held that permanent injunctions do not automatically follow a finding of patent infringement, the *Salinger* court reasoned that nothing in the text or logic of the *eBay* decision limited the rule to patent cases. The *Salinger* court noted that the Supreme Court had expressly relied on copyright cases in reaching its decision and had cited similar provisions in both the Copyright Act and Patent Act for providing injunctive remedies. *Salinger*, 607 F.3d at 78. Further relying on *Winter*, the Second Circuit held that, in the *eBay* decision, the same broad principles of equity applied with equal force to preliminary injunctions as they did to permanent injunctions. *Id.* at 78–79.

Showing Irreparable Harm in the Open-Source Context
In courts where the presumption of irreparable harm no longer applies, open-source plaintiffs will face different challenges in demonstrating the level of injury needed to satisfy the preliminary injunction standard than plaintiffs in traditional copyright cases face. The traditional copyright model reflects an exchange of copyrighted goods for money. Open-source software developers customarily distribute software for free, subject to the terms of an open-source license.

One source of harm that may suffice is that suffered by an open-source developer who builds a reputation and user base by releasing software under the terms of an open-source license and later sells other software products capitalizing on this reputation. This approach helps attract new users who may be reluctant to purchase software developed by a company that lacks an established reputation. When a copyright infringer “borrows” part of the open-source code into a competing product without following the license terms, the infringer detracts from the name recognition and business goodwill built up by the open-source developer. The extent of this loss is difficult to calculate because the open-source developer loses more than just market share. In open-source communities, users add value by directly improving the developer’s software product. End users discriminate among software choices based on the quality of the software, including these improvements. Releasing software under a public license, therefore, has the potential of providing the author with a significant market advantage.

Another source of harm that may suffice to meet the new heightened standard is the harm suffered by open-source developers who distribute open-source licensed software freely but charge for support and warranty services. In this context, the developer’s revenue is directly proportional to the size of the user base. A competitor who incorporates the open-source software into a proprietary product in violation of the license terms reduces the open-source developer’s market share for providing support services. As these lost sales may be difficult to measure, courts may resort to injunctive relief to compensate such plaintiffs.

But what should courts do with the open-source software developer who does not follow a for-profit model? Given the benefits of open-source development, such “pure” open-source developers may have an interest in growing a user base. These benefits increase in proportion to the number of contributing users—the greater the number of contributors, the greater the likelihood of innovation and improvements on the original open-source product. Open-source plaintiffs should be prepared to come forward with admissible evidence supporting their demonstration of harm.

A legal trend that might benefit such open-source plaintiffs is the “sliding-scale” approach to preliminary injunctions. Under the sliding-scale approach, “the greater the likelihood of success on the merits, the less net harm the injunction must prevent in order for preliminary relief to be warranted.” *Judge v. Quinn*, 612 F.3d 537, 546 (7th Cir. 2010). Therefore, if the plaintiff shows that he or she is very likely to prevail on the merits, the court may reduce the magnitude of harm that the plaintiff must show. After making a particularly strong showing of copyright infringement, an open-source plaintiff might be able to convince the court to issue an injunction even if the likely irreparable harm is small in amount. Theoretically, this could overcome the tendency of courts to recognize the motivation for profit favorably against the desire to keep software source code open to the world. Unfortunately for plaintiffs, this approach can be impractical given the difficulty of making a strong showing of infringement based on code buried in proprietary software. Courts that have removed the presumption of irreparable harm are also less likely to allow plaintiffs to prevail on an argument that essentially creates the same result.

Conclusion

The Federal Circuit decision in *Jacobsen I* opened the door for open-source plaintiffs to obtain preliminary injunctions for copyright infringement. The emerging trend of removing the presumption of irreparable harm for copyright cases threatens to close that door. Although it is not clear that the Northern District of California's decision in *Jacobsen II* rejecting the presumption of irreparable harm will become the prevailing rule of law, other courts, including the Second Circuit, have already come to similar conclusions.

Where the presumption of irreparable harm no longer applies, showing such harm and obtaining a preliminary injunction will be difficult for open-source plaintiffs. The plaintiffs with the strongest cases for injunctions will still be those who use the open-source model to build their reputation and market share to produce a profit, giving them supporting evidence of irreparable harm. "Pure" open-source developers who benefit from the distribution of open-source software by itself will have the most trouble. While the sliding-scale approach may benefit such plaintiffs who can make a strong showing of infringement, they must still show that irreparable harm is likely to occur. If the mere fact that the open-source developer has lost the right to control the distribution and modification of his or her software is not enough to establish irreparable harm, then obtaining a preliminary injunction for such "pure" open-source developers may be nearly impossible.

Keywords: litigation, intellectual property, preliminary injunction, open-source license, irreparable harm

S. Christian Platt is a partner, Bob B. Chen is an associate, and Kenneth Newton is a summer associate with Paul, Hastings, Janofsky & Walker, LLP.

The Impact of Recent Case Law on Copyleft Agreements

By R. Scott Rhoades and Jon Rastegar – September 20, 2011

Open-source software is one of the fastest-growing areas in the software industry. In the past decade, both the number of open-source projects and the total lines of open-source code have grown exponentially. Amit Deshpande & Dirk Riehle, "The Total Growth of Open Source," *In*

Proceedings of the Fourth Conference on Open Source Systems (OSS 2008). In addition, the number of corporations employing open-source software has grown dramatically. A recent survey found that 85 percent of companies employ open-source software in their business. David Meyer, *Gartner: 85 Percent of Companies Using Open Source*, cnet. This rapid growth led the Federal Circuit Court of Appeals to note that "open-source licensing has become a widely used method of creative collaboration that serves to advance the arts and sciences in a manner and at a pace that few could have imagined just a few decades ago." *Jacobsen v. Katzer*, 535 F.3d 1373, 1378 (Fed. Cir. 2008). This substantial growth is in part attributable to the open-source community's increased use of reciprocal licensing agreements, sometimes referred to as copyleft agreements.

Copyleft agreements are used to control the availability of open-source software. The agreements are designed to allow a copyright holder to make a software program available to the public for no fee, while also requiring all modified and extended versions of the program to be freely available to the public. The intent of copyleft agreements is to prevent an individual from converting a freely available open-source program into proprietary software.

One of the most common copyleft agreements is the GNU General Public License (GPL), which was devised by the Free Software Foundation. Authors who distribute their works under the GPL "authorize not only copying but also the creation of derivative works," while prohibiting the user from "charging for the derivative work." *Wallace v. Int'l Bus. Machs. Corp.*, 467 F.3d 1104, 1105 (7th Cir. 2006). The GPL allows users to "make and distribute derivative works if and only if they come under the same license terms as the original work." *Id.*; *see* GNU General Public License, § 5, "Conveying Modified Source Versions." "The GPL propagates from user to user and revision to revision: neither the original author, nor any creator of a revised or improved version, may charge for the software or allow any successor to charge." *Wallace*, 467 F.3d at 1105. While the GPL is not the only copyleft agreement available, some commentators estimate that 65 to 70 percent of all open-source software is licensed under the GPL. *See* Sapna Kumar, "Enforcing the GNU GPL," 2006 *U. Ill. J.L. Tech. & Pol'y* 1, 1 (2006).

Despite the growing frequency of the GPL and open-source software, there are surprisingly few cases that analyze copyleft agreements. This lack of analysis by the courts has led some to speculate whether copyleft agreements, like the GPL, effectively accomplish the goal of making derivative works available on the same terms as the original work. *See, e.g.*, Kumar, *supra*, at 1. Critics point out that copyleft agreements like the GPL are unenforceable for a myriad of reasons—ranging from arguments that copyleft agreements amount to price-fixing schemes to arguments that copyleft agreements are unlimited licenses. *Jacobsen* and *Wallace*, the two recent appellate decisions previously mentioned, address these criticisms and resolve some of the uncertainty surrounding the GPL and similar copyleft agreements.

One of the more creative attacks raised against copyleft agreements was the argument that the agreement violated the Sherman Act because it amounted to a price-fixing scheme to "eliminate competition in the operating system market." *Wallace*, 467 F.3d at 1106. In *Wallace*, a software developer alleged that he could not compete in the open market if software was still available free of charge. The developer further alleged that "nothing could be a more effective deterrent to competition." *Id.* The Seventh Circuit required only a "quick look" before strongly rejecting the developer's claims, stating that "the GPL and open-source software have nothing to fear from the antitrust laws." *Id.* at 1108. The strong words of the Seventh Circuit have seemingly put to rest the criticism that a copyleft agreement could constitute an antitrust violation.

Another criticism of copyleft agreements looks at whether a copyright infringement claim may be brought against a software provider who violates the agreement. Specifically, the issue is whether the underlying copyright in the software remains enforceable; that is, whether a copyright holder can use a copyleft agreement to dedicate certain work for public use and still enforce the copyright. A copyright holder cannot pursue a copyright-infringement claim against someone who is acting within the scope of his or her license. Thus, if the scope of the license is limitless, the copyright holder could not bring an infringement claim against a licensee. However, if the scope of the nonexclusive license is limited, actions beyond the license limits would be copyright infringement. With regard to copyleft agreements, the scope of the license granted by the copyright holder must be determined. Typically, a determination that a limited scope has been granted to the licensee turns on whether the specific provisions of the license are conditions or merely covenants. *See S.O.S., Inc. v. Payday, Inc.*, 886 F.2d 1081, 1087 (9th Cir. 1989). If the provisions of the license are conditions and the user violated those provisions, then the user is outside the scope of the license and the copyright holder can sue for copyright infringement. *Id.* On the other hand, if the provisions are merely covenants, then the copyright holder did not limit the license scope; therefore, the only remedy available to the copyright holder is a breach of contract claim. *See Graham v. James,* 144 F.3d 229, 236–37 (2d Cir. 1998).

The Federal Circuit recently addressed this issue, holding that failure to comply with the attribution and modification transparency requirements in an open-source software license could constitute copyright infringement. *See Jacobsen*, 535 F.3d at 1382. In *Jacobsen*, a copyright holder sued a software developer for copyright infringement based on the developer's failure to abide by the terms of an open-source licensing agreement. The agreement granted users the right to copy, modify, and distribute the software

> provided that [the user] insert a prominent notice in each changed file stating how and when [the user] changed that file, and provided that [the user] do at least ONE of the following:
>
> > a) place [the user's] modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include [the user's] modifications in the Standard Version of the Package.
> > b) use the modified Package only within [the user's] corporation or organization.
> > c) rename any non-standard executables so the names do not conflict with the

> > standard executables, which must also be provided, and provide a separate manual page for each nonstandard executable that clearly documents how it differs from the Standard Version, or
> > d) make other distribution arrangements with the Copyright Holder.

Id. at 1380.

The district court held that the scope of the license was "intentionally broad" and that the above attribution and modification requirements did not "limit the scope of the license." *Jacobsen v. Katzer*, 2007 WL 2358628 at *7 (N.D. Cal. 2007), *rev'd*, 535 F.3d 1373 (Fed. Cir.). Accordingly, the district court held that because the scope of the licenses was unlimited, the user was necessarily within the scope of the licensing agreement and the attribution and modification requirements could only be viewed as covenants. Based on this reasoning, the court held that the copyright holder could not recover for the alleged copyright infringement.

The Federal Circuit reviewed the copyleft licensing agreement and reversed the district court's holding, stating that the contract's intent was to create a condition rather than a covenant. 535 F.3d at 1382. To support this conclusion, the Federal Circuit cited the use of the phrase "provided that" in the licensing agreement and noted that, under the relevant state law, such a phrase typically denoted a condition. *Id.* at 1381. Ultimately, the Federal Circuit held that the terms of the copyleft licensing agreement were "enforceable copyright conditions," not covenants, and for that reason, the agreement could support a copyright infringement claim. *Id.* at 1383.

It should be noted that the Federal Circuit's holding in *Jacobsen* does not stand for the proposition that all open-source licensing agreements allow a copyright holder to bring a claim for copyright infringement based on failure to comply with the agreement. Rather, *Jacobsen* supports the proposition that an open-source licensing agreement can have conditions that limit the scope of the license and that a copyright holder can bring a claim for copyright infringement based on actions outside the scope of the copyright license. *Id.*; *see also Nimmer on Copyright* § 1015[A] (1999).

Whether the scope of a particular agreement has limits depends on the precise language in the agreement and the applicable state law. For example, the GPL provides as follows:

> [A user] may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, *provided that [the user] also meet all of these conditions*:
>
> > a) The work must carry prominent notices stating that [the user] modified it, and giving a relevant date.
> > b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the

> requirement in section 4 to "keep intact all notices."
> c) [The user] must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if [the user] have separately received it.
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, [the user's] work need not make them do so.

GNU GPL, § 5, "Conveying Modified Source Versions" (emphasis added).

While the GPL has yet to be interpreted by a Federal Court, it seems likely that the phrase "provided that [the user] also meet all of these conditions" places a condition, and not a covenant, on the user. Therefore, following the Federal Circuit's reasoning in *Jacobsen*, the provisions of the GPL likely limit the scope of the copyright license and allow a copyright holder to pursue copyright-infringement claims for actions beyond that scope.

While the law surrounding copyleft agreements is still in its infancy, the recent appellate court decisions in *Wallace* and *Jacobsen* suggest that copyleft agreements may be effective in ensuring that copies and modifications to open-source software will remain open source. In these cases, the courts have clearly taken note of the public benefits provided by open-source software. The courts' acknowledgement of the sound public policy behind open-source software suggests that the courts will continue to look favorably on open-source licensing and that copyleft agreements will remain effective and enforceable.

Keywords: litigation, intellectual property, copyleft agreements, GNU General Public License

R. Scott Rhoades is a senior counsel and Jon Rastegar is an associate at Akin Gump Strauss Hauer & Feld in Dallas, Texas.

Vendor Indemnification on the Open Range

By David Swetnam-Burland and Stacy O. Stitham – September 20, 2011

When it comes to intellectual property protection, do businesses get what they pay for when they use open-source software? Cost considerations and the ability to tinker at will with licensed software may make open source an attractive option for any company in the market for an operating system, web server, or software —as may the romantic notion that software development should be an open range for exploration and innovation. However, for an online business defending a patent infringement claim based on its use of an open-source product to develop its website, the open-source solution may look less attractive with each passing settlement demand or bill for attorney fees.

By no means do we intend to suggest that the open-source solution is a bad choice for a cost-conscious, growing operation; indeed, there are many reasons to recommend it. Nor do we have any desire to take a position on the comparative merits of open-source software as contrasted with software solutions offered by established vendors that charge for their services. However, for any company weighing its options, we do hope to prompt careful consideration of the intellectual property ramifications of adopting open-source technology. As e-commerce firms become a growing target of increasingly expensive patent lawsuits, open-source software may come with costs that don't appear on the label.

As-Is Software

Open-source software is commonly defined by the 10 criteria listed on the website of Open Source Initiative at http://opensource.org/docs/osd (last visited May 25, 2011), but it boils down to just what the name implies—an opening of the source code to one and all, allowing distribution and redistribution that includes, in most instances, modifications and derivations under the same terms as the original license. The relatively free-ranging distribution mechanism—which allows users to adopt source code under an existing license that was created without any input from current or potential users for code that may or may not have been modified and software that the user can implement alongside other applications—is problematic from the standpoint of intellectual-property protection because the very freedom of this "free" software breeds legal uncertainty that can cause headaches for the user long after adoption.

While open-source software licenses come in all shapes and sizes, they share at least one thing in common. Using software developed by another brings exposure to potential allegations of intellectual-property infringement, which increasingly involve patent-infringement claims relating to e-commerce activities. While any software license—proprietary or open source—carries with it some risk of an infringement claims, open-source licenses more frequently come packaged "as is" with respect to indemnification for third-party claims and warranties of title. For instance, both Red Hat, Inc., and the Apache Software Foundation, two of the major players in the open-source field, include disclaimers of warranties strongly suggesting that, should a patent owner come calling with a demand letter or complaint of infringement, it is the licensee who may be left to answer the door and the patent owner's questions.

Red Hat's Enterprise Agreement contains not only a limitation of liability and disclaimer of damages, but also a disclaimer of warranty that reads, in relevant part, as follows:

> **10.2 Disclaimer of Warranty. EXCEPT AS EXPRESSLY PROVIDED IN SECTION 10.1 OR BY A THIRD PARTY VENDOR DIRECTLY TO CLIENT UNDER A SEPARATE AGREEMENT, THE SERVICES, SOFTWARE AND ANY HARDWARE ARE PROVIDED BY RED HAT "AS IS" AND WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, INCLUDING THE IMPLIED**

> **WARRANTIES OF MERCHANTABILITY, NON–INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. RED HAT DOES NOT GUARANTEE OR WARRANT THAT THE USE OF THE SERVICES, SOFTWARE OR HARDWARE WILL BE UNINTERRUPTED, COMPLY WITH REGULATORY REQUIREMENTS, BE ERROR FREE OR THAT RED HAT WILL CORRECT ALL SOFTWARE ERRORS.**

Red Hat, *Red Hat Enterprise Agreement* (last visited May 25, 2011).

Apache's current license is only slightly less emphatic:

> **7. Disclaimer of Warranty**. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON–INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

Apache Software Foundation, *Apache License, Version 2.0* (last visited May 25, 2011).

With such take-it-as-it-comes language governing the incorporation of open-source products and services into a business, it is worth a moment's pause prior to implementation to consider that the thriving e-commerce operation or other web-based operation of a business may someday be confronted with a patent-infringement lawsuit based on its use of an Apache HTTP server or a Red Hat Linux operating system. What's more, there is a distinct possibility that neither Apache nor Red Hat will ride to the rescue of a user with whom they may never have had any direct contact. It appears that open-source providers are becoming aware of this concern, as Red Hat now offers customers with valid subscriptions an "Open Source Assurance"—including a promise that it will defend and indemnify such customers—which it makes available to subscribers under a separate "Open Source Assurance Agreement." Red Hat, *Open Source Assurance* (last visited August 11, 2011).

Because software modification by the end user is not only possible but encouraged by the nature of open-source licensing and use, and because open-source products will likely be used along with a host of other software or computer products, there is a strong likelihood that, when it comes time to figure out who will be responsible for the costs of defense, damages, or both, the buck will stop with the e-commerce business. As demonstrated by a recent Supreme Court opinion and two recent orders of the Federal Circuit Court of Appeals, the law of patent infringement based on the conduct of multiple actors is evolving rapidly, which leaves businesses that are contemplating open-source software in a double bind. By choosing an open-source solution, they may be "purchasing" potential patent infringement liability without vendor protection in a legal landscape that is changing by the minute and in not entirely predictable ways. *See Global-Tech Appliances, Inc. v. SEB S.A.* [PDF], No. 10–6 (May 31, 2011) (induced infringement); Order, *Akamai Techs., Inc. v. Limelight Networks, Inc.* [PDF], 2009-1372, -1380, -1416, -1417 (Fed. Cir. Apr. 20, 2011) (ordering rehearing en banc on joint infringement standard); Order, *McKesson Techs., Inc. v. Epic Sys. Corp.* [PDF], 2010–1291 (Fed. Cir. May 26, 2011) (same).

Of course, a proprietary license is no saving grace. But with a provider of proprietary software, at least the customer may have a better chance of negotiating a license face-to-face with a vendor and securing more favorable indemnification language up front as part of the purchase and sale process, especially if the customer is a business with some size or clout in its industry. Unlike the decentralized open-source software, where there is often no single authority or locus of responsibility for any given product, proprietary software usually has a clearer ownership structure and pedigree, as well as a client representative designated to take the customer's calls when a demand letter or lawsuit falls into the customer's lap. That said, depending on the size of the business or the type of product it is purchasing, the business may have no more leverage in negotiating a proprietary software license than it would an open-source license. The only way a business can reach an informed decision on this point is to consider the issue beforesigning up for the software or service.

Case Study
In 2004, in connection with its consideration of insurance for open-source products, the insurance firm Open Source Risk Management (OSRM) conducted an analysis of the risks posed by Linux and concluded that it potentially infringed 283 patents. Press Release, Open Source Risk Mgmt., *Results of First-Ever Linux Patent Review Announced, Patent Insurance Offered by Open Source Risk Management* [PDF] (Aug. 2, 2004). At the time, the identified patents had not been litigated and the claims within them not yet tested against Linux end users. Seven years later, however, the insurance industry's cries seem more prescient than pessimistic. One month before this article was written, a jury in a federal court in the Eastern District of Texas awarded Bedrock Computer Technologies, Inc., millions in damages for Google's infringement of a Linux kernel patent. Bedrock has sued Linux users, from AOL to Yahoo!, and there are fears that the Google verdict will have reverberations for other Linux end users. This is not to say that distributor Red Hat has sat on its hands while its flagship product has been maligned; indeed, it has sued Bedrock seeking to invalidate the patent in suit. But, as in similar e-commerce disputes, patent holders are demonstrating a willingness to target the successful users of open-source software, and there is no certainty that open-source distributors can or will step up to defend their products in every case or in cases involving software that has undergone a set of transformations over time.

Sunnier Skies
As noted at the outset of this article, however, we have not come to bury the open-source model but rather to point out a potentially hidden cost of the freedom that comes with such a model. For

the enterprising business looking to adopt open-source software yet is still concerned about the risks of intellectual-property-infringement actions, all is not doom and gloom. Careful examination and assessment as well as pre-adoption of the likely risks of any particular open-source product under consideration (such as the Apache web server or Linux operating system) will help any business make an informed decision about which software products to integrate into its operations.

And should a company find itself with an open-source solution on hand that arrived prepackaged with a hidden, and unwelcome, side of litigation, the first avenue of potential relief may still be the specific vendor or distributor of the product. Assuming there are deep enough pockets and a strong enough motive, such a vendor may be prevailed upon to step forward and defend its product (à la Red Hat in the *Bedrock* litigation). Open-source purveyors have a strong business interest in defending their business model and in showing the business world that they have nothing to fear from opening their operations to open-source software or services. Customers of sufficient size or in sufficient numbers may be able to make a strong case to open-source firms that defending customers is good business in a market where less tech-savvy businesses might otherwise see proprietary software licensing as a safer bet.

Furthermore, open-source software customers have the option of purchasing intellectual-property insurance through third-party vendors. For example, OSRM claims on its website that it is “the exclusive risk assessor on the world’s first insurance facility to cover the specialized risks faced by enterprises that include or rely upon elements of Linux and other Open Source software in their commercial products or IT infrastructure.” *See Open Source Insurance*, Open Source Risk Management (last visited May 31, 2011).

Finally, it is worth noting that the very factor that makes open source so difficult to pin down when it comes to warranties—its mutable and modifiable source code—may be useful in permitting (or even encouraging) design-arounds of patent-infringement claims, thereby minimizing damages in cases in which litigation is inescapable.

Conclusion

Like everything else, open-source software carries both pros and cons; and the extent to which a business decides to integrate or use open-source products in its operation remains, in essence, a business assessment. With up-front consideration of the potential negative intellectual-property ramifications of adoption—including analysis of license terms and product risk, assessment of the vendor in question, and consideration of whether or not to purchase insurance—a business can be sure that it is walking into a licensing decision regarding the merits of open-source software with open eyes.

Keywords: litigation, intellectual property, open-source software, patent infringement

David Swetnam-Burland and Stacy O. Stitham are partners at Brann & Isaacson, in Lewiston, Maine.

Intellectual Property Litigation
FROM THE SECTION OF LITIGATION INTELLECTUAL PROPERTY LITIGATION COMMITTEE

Fall 2011, Vol. 23 No. 1

Leveraging Open-Source Software in Patent Litigation

By Andrew Strickland and Amy Chun – September 20, 2011

In the context of software-patent litigation, open-source licenses are generally an afterthought during discovery, if they are considered at all. By failing to consider their opponents' use of open-source licenses, however, accused infringers may be missing out on opportunities to discover key facts that could open the door to new litigation strategies. For example, there may be situations where an accused infringer may be able to limit damages, raise new defenses, or file counterclaims because an opponent licensed software under an open-source license or used open-source software. This may be especially true when the open-source license in question is the GNU General Public License (GPL) version 3, or GPLv3.

The GNU GPL is one of the most common licenses for open-source software. The most recent version of the GPL, GPLv3, contains express provisions related to patent licensing. For example, licensees are granted a royalty-free patent license to "essential patent claims" of the licensor—patent claims that would be infringed by making, using, or selling the software if there were no license in place. In addition, licensees may not impose a license fee, royalty, or other charge for patent rights granted by the licensor to downstream recipients under GPLv3. Other provisions prevent licensees from engaging in practices pertaining to discriminatory licenses.

In addition to the patent-specific provisions, another attribute of GPLv3 that may interest patent litigators is that it is a "viral" software license because new software derived from GPLv3 source code (GPLed software) must be distributed under the GPLv3 to comply with the license. The viral nature of GPLv3 is the heart of the bargained-for exchange of the license; the licensor allows the licensee to use, modify, or redistribute its GPLed software, provided the licensee also licenses any distribution of it under GPLv3.

While some software companies understand that GPLv3 is viral and intentionally incorporate GPLed software into their products, this is not always the case. For instance, a developer may be tasked with developing software that will be released under a proprietary license (that is, one that would not allow licensees to modify and distribute the source code). Unbeknownst to the company, the developer incorporates GPLv3 source code into the company's software. Using this GPLed source code is permitted under GPLv3 (perhaps even encouraged), but only if the licensee abides by the terms of the license and releases its software under GPLv3. Here, when the company releases its software under a proprietary license without knowing the software includes GPLv3 source code, it has violated GPLv3.

Although use of open-source software like GPLv3 is rapidly increasing, issues arising from the patent provisions of GPLv3 have yet to appear in a published court opinion. Therefore, while it is not clear how courts will treat GPLv3's patent provisions, the provisions are worth exploring because they could affect how either a plaintiff (patentee) or a defendant (accused infringer) litigates a case. For example, counsel for plaintiffs should be aware of potential risks or pitfalls that may arise from their client's own use of GPLv3 or GPLed software. Counsel for defendants, on the other hand, should be looking for opportunities to learn of the plaintiff's use of GPLv3 or GPLed software and leverage that use to maximize defensive strategies.

A hypothetical fact pattern helps illustrate the potential patent-litigation strategies pertaining to GPLv3. Suppose a software company named MegaSoft holds a large patent portfolio containing claims that cover many of its profitable consumer software products. One of those patented products is a tax-preparation program, MegaTax, which allows small businesses the ability to file their taxes. MegaSoft learns that GratisSoft began distributing an open-source tax-preparation program, GratisTax, that has the ability to read and modify files in MegaSoft's proprietary, patented file format claimed in MS Patent I. MegaSoft files suit against GratisSoft for infringement. GratisSoft's counsel conducts an initial analysis and realizes that the prognosis is not good: MS Patent I's claims read on GratisSoft's activities, and it is likely that a court will find MS Patent I valid and infringed by GratisSoft. Faced with this bad news, GratisSoft's counsel starts brainstorming and develops the following additional strategies: limiting damages, finding creative defenses, and discovering hidden counterclaims.

Limiting Damages

GratisSoft may argue that its damages to MegaSoft should be limited if it discovers MegaSoft licensed software that embodies the claims of MS Patent I under GPLv3. Under GPLv3, the licensor grants a license for a patent that reads on the software being licensed. Accordingly, if MegaSoft licensed software under GPLv3 that reads on MS Patent I, then it will have granted a royalty-free patent license to another party to make, use, or sell MS Patent I. GratisSoft may leverage this fact to try to limit its damages to MegaSoft because existing patent licenses are an important factor when determining damages under a lost profits or reasonable royalty theory.

Lost Profits

Lost profits may be available to a plaintiff patentee when the plaintiff can show that it lost sales it would have made but for the defendant's infringing activity. It may be possible for a defendant to attack a claim of lost profits by showing there were free, licensed products on the market, and GPLv3 could be used as a tool to demonstrate the existence of such products.

Suppose, in addition to the hypothetical facts presented above, MegaSoft intentionally licensed software that embodied MS Patent I's claims under the GPLv3 in its own product that reads MegaTax files called MegaTaxReader. Under the GPLv3, a licensee of MegaTaxReader could redistribute the source code without infringing MS Patent I's claims, and anyone receiving that distribution would also receive a royalty-free license to practice MS Patent I's claims. Suppose further that a third party improves MegaTaxReader by building a tax-preparation program around it called FreeTax that is similar to MegaTax but is offered for free. Because the third party is a downstream recipient of MegaSoft's GPLv3 distribution of MegaTaxReader, the third party has a license to practice MS Patent I and FreeTax does not infringe MS Patent I. Accordingly, there are three tax-preparation software products on the market: MegaTax, FreeTax, and GratisTax. GratisSoft may now argue that MegaSoft cannot prove that it lost sales for all of the GratisTax products because MegaSoft's release of MegaTaxReader under GPLv3 enabled the development of FreeTax, a free competing product. Thus, if GratisTax was not available, at least some of the GratisTax customers might have chosen FreeTax rather than MegaTax. Moreover, even though FreeTax has a license under MS Patent I, the license is royalty-free. GratisSoft can argue that MegaTax would not have lost profits if the GratisTax customers chose FreeTax.

This example stresses the importance of drafting discovery requests that cover the use of GPLv3. If GratisSoft asked MegaSoft only for licenses that covered the claims of MS Patent I, the MegaSoftReader license (GPLv3) would likely not have been discovered because it does not specifically name MS Patent I. GPLv3 grants a license for "essential patent claims," not for the claims of any specifically named patent. Thus, patent litigators should consider drafting discovery requests directed to identifying open-source distributions of technology similar to the patent-in-suit.

Reasonable Royalty

When a plaintiff cannot prove lost profits, the plaintiff may still receive damages in the form of a reasonable royalty. Courts frequently define a reasonable royalty as a royalty that would have resulted from a hypothetical negotiation between a willing patent owner and a willing potential user. To determine a reasonable royalty, courts consider several factors, which include, among others:

- the royalties received by the plaintiff patentee for licensing the patent, tending to prove an established royalty;
- the plaintiff's established policy and marketing program to maintain a patent monopoly by not licensing others to use the invention or by granting licenses under special conditions designed to preserve that monopoly;
- the established profitability of the product made under the patent, its commercial success, and its current popularity; and
- the amount that a licensor (such as MegaSoft) and a licensee (such as GratisSoft) would have agreed upon at the time the infringement began if both had been reasonably and voluntarily trying to reach an agreement.

Because courts give considerable weight to the first factor, GratisSoft could possibly limit MegaSoft's reasonable royalty rate by arguing that the free distribution of MegaSoftReader shows a royalty of zero for at least some of its licenses of MS Patent I.

It is important to note, however, that nothing prevents a licensor from charging a fee when it distributes software under GPLv3. The licensor need only distribute, or make available, the source code with the distribution. Thus, if MegaSoft charged for MegaSoftReader, it could argue that the royalty is not zero. GratisSoft, however, can point out that the GPLv3 states, "Each [licensor] grants you a non-exclusive, worldwide, royalty-free patent license under the [licensor's] essential patent claims." Thus, the license is explicit: Patent rights are granted royalty-free. Accordingly, GratisSoft could argue that any money MegaSoft received for MegaSoftReader is for other rights associated with the software, or it could be a distribution or service charge, as opposed to a royalty in exchange to practice MS Patent I. However, MegaSoft may counter by arguing that it provided other benefits, consideration, or a combination of these, which entitles it to a royalty rate greater than zero.

Further, the other factors listed above may present additional points of argument for GratisSoft to limit the reasonable royalty rate. For example, because MegaSoft offered MegaTaxReader under GPLv3, GratisSoft could argue that MegaSoft failed to maintain its patent monopoly and, due to the viral nature of GPLv3, encouraged others to practice the claims of MS Patent I because downstream recipients (who may have no direct contact with MegaSoft) would hold a license to MS Patent I’s claims. In addition, GratisSoft could argue that the profitability of the products made under MS Patent I is limited. While MegaSoft may generate revenue from MS Patent I, it would not receive revenue from downstream licensees of MS Patent I’s claims under the provisions of GPLv3.

Finally, in a hypothetical, arm’s length negotiation between GratisSoft and MegaSoft, GratisSoft might argue that it has little incentive to pay a royalty for a license to use MS Patent I. Because the GPLv3 grants licensees the right to redistribute the licensed software, GratisSoft may argue that it could have obtained the source code to MegaTaxReader for free or for a nominal fee. In addition, GratisSoft may argue that because MegaSoft would not receive any royalties or other payments from downstream licensees, MegaSoft did not intend to receive royalties for MS Patent I and, therefore, would be likely to accept a royalty rate of zero in a negotiation.

The Effect of GPLv3’s Viral Provisions on Damages Arguments
The damages examples above focused on a situation where MegaSoft had intentionally licensed software under GPLv3 and then later asserted a patent against GratisSoft, which independently developed GratisTax. Because GPLv3 is viral, even if MegaSoft did not intentionally release MegaTaxReader under GPLv3 (or did not release MegaTaxReader at all), GratisSoft may have other arguments (albeit weaker ones) limiting its damages if MegaSoft inadvertently incorporated GPLv3 software in its code.

Suppose, for example, that during development, a programmer working for MegaTax downloaded some GPLv3 source code, modified it, and then introduced the modified version into the code base of MegaTax. Because MegaTax is not released under GPLv3, MegaSoft is in violation of GPLv3. If GratisSoft discovers this fact, it may argue that, while MegaSoft did not license MegaTax under GPLv3, it should have done so and is bound by its terms. Accordingly, GratisSoft could then possibly make the arguments outlined above with respect to lost profits and reasonable royalty.

MegaSoft may point to GratisSoft’s reliance on the viral provision to attack those arguments. First, MegaSoft might argue that GratisSoft should not be able to impose the terms of GPLv3 on MegaSoft because GratisSoft was not the licensor of the GPLed software that was added to the code base of MegaTax. In other words, MegaSoft would argue that GratisSoft does not have standing to enforce the GPLv3 provision. Second, MegaSoft could also argue that because the inclusion of GPLed software was inadvertent, the inclusion does not establish a royalty because it was not part of MegaSoft's plan to enforce MegaSoft Patent I.

Defenses

Implied License under Legal Estoppel

One tool an accused infringer may use to try to absolve itself of liability for infringement is an implied license. An implied license permits the accused infringer to practice the claims of a patent without the express permission of a patentee. One theory supporting an implied license is legal estoppel.

Legal estoppel applies when a patentee attempts to enforce against an accused infringer a first patent that must be practiced for the accused infringer to gain the benefit of a license for a second patent granted by the patentee. The policy behind the legal estoppel theory is that once a patentee licenses a property right in a patent, it cannot detract from that right through the use of another patent. A recent case, *TransCore, LP v. Elec. Transaction Consultants*, 563 F.3d 1271 (Fed. Cir. 2009), from the Court of Appeals for the Federal Circuit, illustrates an example of an implied license through legal estoppel.

In *TransCore*, the patentee (TransCore) attempted to enforce the '946 patent against an accused infringer (ETC). The infringement allegation stemmed from ETC's use of a product it purchased from a third party (Mark IV). Before TransCore's suit against ETC, it reached a settlement agreement with Mark IV that contained a covenant not to sue Mark IV for infringement of the '082 patent. Although the broader '946 patent was not part of the settlement agreement (it issued after the settlement agreement was reached), in order to practice the claims of the '082 patent, Mark IV would necessarily have to practice the claims of the '946 patent. The Federal Circuit determined that TransCore was legally estopped from asserting the '946 patent against Mark IV. Because ETC obtained its product from Mark IV, TransCore's patent rights were exhausted, and TransCore could not enforce the '946 patent against ETC.

To expand on this example, suppose MegaSoft distributed a product to a third party that is covered by a different patent, MS Patent II, under GPLv3, and suppose the third party modified the product and released the modification under GPLv3 as ThirdTax, which GratisSoft then modified and used in GratisTax. MegaSoft then obtained MS Patent I, which covers related subject matter to, and is broader in scope than, MS Patent II. MS Patent II cannot be practiced without necessarily infringing MS Patent I.

Here, although GratisSoft conducted no direct transactions with MegaSoft, GratisSoft may argue that it has an implied license under *TransCore*. GratisSoft received the software covered by MS Patent II from the third party and has rights to practice MS Patent II under GPLv3. Because GratisSoft cannot exercise its right under GPLv3 without necessarily infringing the claims of MS

Patent I, legal estoppel may apply to relieve GratisSoft of its liability for infringement of MS Patent I.

Challenging Inventorship
Another tool available to defendants is the ability to challenge the inventorship of a patent under 35 U.S.C. § 102(f), which states that a person shall be entitled to a patent unless "he did not himself invent the subject matter sought to be patented." One of the features of GPLv3 is the distribution of source code allowing programmers to modify the code for their own purposes and redistribute it as part of their own products. If an accused infringer learns that a portion of the patentee's patented software contains source code originally licensed under GPLv3, and the patent claims cover features from the GPLv3 code, the accused infringer may be able to challenge the inventorship of the patent. Of course, the ability to leverage section 102(f) depends on the nature of the use of the GPLv3 source code.

For example, if MegaSoft used only a minor portion of GPLed software to develop the functionality that became claimed subject matter in MS Patent I, then a section 102(f) challenge by GratisSoft may not be appropriate. If, however, the claims of MS Patent I contain subject matter that originated in the GPLed software, and MegaSoft did not name the appropriate inventor, GratisSoft may be able to invalidate the patent due to improper inventorship.

Finding Hidden Counterclaims
Perhaps the most straightforward strategy a defendant might employ to leverage a plaintiff's use of GPLv3 software is for the defendant to terminate a patent license it granted to the plaintiff under GPLv3 because of a violation of GPLv3's terms. This technique may be useful in situations where the plaintiff may not be aware that the defendant's GPLed source code ended up in its product.

For example, suppose GratisSoft owns a small patent portfolio that includes GS Patent, which covers a new method for suggesting misspelled words. GratisSoft developed GratisSpellChecker, which embodies the claims of GS Patent. GratisSoft discovers that MegaSoft used GratisSpellChecker in its word processing program, MegaWord, which is very profitable and is not released under GPLv3. GratisSoft now has a counterclaim against MegaSoft. Because MegaWord is not licensed under GPLv3, it is in violation of the viral provision of the license. And because GPLv3 is violated, GratisSoft's royalty-free license to MegaSoft under GS Patent terminates, and MegaSoft's making and selling of MegaWord are an infringement of the GS Patent. MegaSoft's infringement now provides GratisSoft with a bargaining chip in settlement negotiations.

It is interesting to note that, in the above example, GratisSoft may not have learned of the GPLv3 violation without MegaSoft filing suit and without GratisSoft expanding discovery to products other than MegaTax. MegaWord may have been offered only as an off-the-shelf product, and GratisSoft may have had no reason to believe MegaSoft was infringing the claims of the GS

Patent. It was only through discovery that GratisSoft learned of MegaSoft's activity providing a basis for a counterclaim.

Conclusion

While the arguments presented above may not be available in all cases, software-patent litigators should not overlook GPLv3 when developing litigation strategy. Given the increasing use of open-source software and the growing number of patent-software cases, there may be additional arguments based on GPLv3 or other open-source licenses. To prepare a robust defense on behalf of their clients, patent litigators ought to consider arguments based on open-source licenses that they may have previously overlooked.

Keywords: litigation, intellectual property, open-source software, GNU General Public License

Andrew Strickland is an associate and Amy Chun is a partner at Knobbe, Martens, Olson & Bear, LLP, in Irvine, California.

Young lawyers may be particularly interested in this article because they often deal with pleading standards.

Open-Source Software in the Cloud

By Bradley J. Walz – September 20, 2011

If you pay attention to the current marketing of some big tech companies, such as IBM and Microsoft, you will notice that more references are being made to "the cloud." This is another indication that cloud computing is not just for businesses anymore. Consumers are also recognizing the benefits that cloud computing can provide. Red Hat's chief executive officer, Jim Whitehurst, said that cloud computing is heavily dependent on open-source software and that 90 percent of today's clouds leverage some open-source software. Tim McElligott, *Cloud Drives Red Hat Back to Asia*, Billing World, Dec. 30, 2010. Accordingly, the issues involved with open-source software are something of which to be aware when working with cloud computing providers and businesses as well as with consumers.

Cloud Computing

There is no statutory definition of cloud computing, but the National Institute of Standards and Technology (NIST), an agency of the U.S. Department of Commerce responsible for defining national standards and working to apply them globally, has defined cloud computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." *The NIST Definition of Cloud Computing* [PDF], NIST. (Dec. 30, 2010).

NIST identifies five essential characteristics of cloud computing:

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed, and can do so automatically without requiring human interaction with each service's provider.
- *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants).
- *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control over or knowledge of the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources are storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity.* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- *Measured service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service, such as storage, processing, bandwidth, and active user accounts. Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of a service.

Cloud computing providers offer their services through different models. The first model is Software as a Service (SaaS). SaaS provides the consumer with the capability to use the cloud provider's software applications running on a cloud infrastructure. The applications are accessible from various devices through a thin interface such as a web browser (for example, web-based email). The consumer does not manage or control the underlying cloud infrastructure, which includes the network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited, user-specific application configuration settings. Examples of SaaS are Salesforce.com and its customer-relationship management application; Microsoft's Business Productivity Online Suite, soon to be revised as Office 365; and Google Apps for email and document collaboration. The code that runs the application is managed by the cloud provider, and the customer simply signs up as a user.

The second service model is Platform as a Service (PaaS). PaaS provides the consumer with the capability to deploy, onto the cloud infrastructure, consumer-created or -acquired software applications created using programming languages and tools supported by the cloud provider. The consumer does not manage or control the underlying cloud infrastructure, including the network, servers, operating systems, or storage, but the consumer has control over the deployed applications and, possibly, application-hosting environment configurations. Google's App Engine is an example of PaaS. The consumer writes the application in Java or Python and uploads it. The consumer's application can make use of services such as a transactional data store, task queue, user management, email, and caching, and the consumer relies on Google to provide these services.

The third service model is Infrastructure as a Service (IaaS). IaaS allows the consumer to access provision processing, storage, networks, and other fundamental computing resources and to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications—and, possibly, limited control over select networking components (such as host firewalls). Amazon dominates the IaaS market with its Elastic Compute Cloud (EC2).

SaaS, PaaS, and IaaS are deployed, in turn, through different models. One deployment model is a private cloud. A private cloud is operated solely for an organization. The organization or a third party may manage it, and it may exist on or off the organization's premises.

Another deployment model is a community cloud. A community cloud is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or by a third party and may exist on premise or off premise.

A third deployment model is a public cloud. A public cloud is made available to the general public or a large industry group and is owned by an organization selling cloud services.

Finally, there is a hybrid cloud. A hybrid cloud comprises two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability.

Open-Source Software

The Free Software Foundation (FSF) is a nonprofit organization with a mission to promote computer user freedom. The FSF identifies four essential characteristics of "free software":

- The freedom to run the program for any purpose;
- The freedom to study how the program works and change it;
- The freedom to redistribute copies of the program; and
- The freedom to distribute copies of the modified program versions to others.

To accomplish a couple of these freedoms, access to the source code is necessary.

There are different kinds of free software, but the distinctions among the categories are minimal. The one category of free software that most concerns software developers is "copyleft software." Copyleft software's distribution terms ensure that all copies of all versions carry the same distribution terms. For example, copyleft licenses generally do not allow others to add requirements to the software (though a limited set of safe added requirements can be allowed), and they generally require making source code available. This shields the program and its modified versions from some of the common ways of making a program proprietary. The FSF publishes the GNU General Public License (GPL), which is the most popular copyleft software license. Other important licenses published by the FSF include the GNU Lesser General Public License (LGPL), GNU Affero General Public License (AGPL), and GNU Free Document License (FDL).

The Intersection Between Cloud Computing and Open-Source Software

Generally, open-source software will have less of an impact on end users of SaaS and IaaS than on end users of PaaS. End users of SaaS only have access to the application; they do not receive a copy of the software in either object code or source code. Accordingly, there is no opportunity to create a derivative work. And the ability to create a derivative work is key to whether proprietary software is exposed to the distribution terms of a copyleft license. End users of IaaS, like end users of SaaS, generally control the operating system and applications. As a result, open-source software has little impact on end users of IaaS. However, end users of PaaS do not control the platform used to deploy a proprietary application. If the platform is subject to a copyleft license, how the application and platform interact with each other could put the proprietary nature of the application in jeopardy because there is the potential to create a derivative work.

In most cases, an operating system is not a behemoth but consists of many small system programs governed by the core, or kernel, of the operating system. An operating system manages computer hardware resources and provides common services for the execution of various applications. The operating system provides a layer of abstraction between the application and the hardware. In other words, applications interact with the hardware through the operating system.

The FSF developed the GNU operating system, which is based entirely on free software. The kernel in the GNU operating system is called the GNU Hurd. However, after more than 25 years of development, the Hurd has not achieved production quality. Steven J. Vaughan-Nichols, *Opinion: The Top 10 Operating System Stinkers*, Computerworld, Apr. 9, 2009. The first version of the Linux kernel ported GNU code, including the GNU C Compiler, to run on Linux. Later, when the GNU developers learned of Linux, they adapted other parts of GNU to run on the Linux kernel. This work filled in the remaining gaps to running a completely free operating system.

Modern free and open-source software systems are composed of software by many different authors, including the Linux kernel developers, the GNU Project, and other developers, such as those behind the X Window System. Desktop and server-based distributions use GNU components such as the GNU C Library (glibc), GNU Core Utilities (Coreutils), and bash. In an analysis of the source code for packages comprising Red Hat Linux 7.1, a typical Linux distribution, the total size of the packages from the GNU Project was found to be much larger than the Linux kernel. David A. Wheeler, *More Than a Gigabuck: Estimating GNU/Linux's Size*, David A. Wheeler's Personal Home Page, July 29, 2002. Determining exactly what constitutes the "operating system" is a matter of continuing debate.

GNU components are licensed under the GNU GPL. The GNU GPL permits an end user to have the free software on the condition that any derivative works the end user creates from it and distributes must be licensed to all others under the same license terms, although the actual language of the GNU GPL does not state the bargain so clearly. Nevertheless, the key elements that would subject proprietary software to the terms of the GNU GPL are the creation of a derivative work and distribution.

Under copyright law, a derivative work is a work based on one or more preexisting works, whereas a mere collection of independent works is a "collective work" and not a derivative work. This distinction is important because it has a bearing on the reach of the GNU GPL and its applicability to individual, unmodified programs that merely link to each other and are collected into one program for convenience.

To form an executable program, various software modules need to link together. A link may be static or dynamic. A static link occurs when some or all of the functionality from an object code module forms part of the final executable program when the source code is compiled into object code. A dynamic link occurs when some or all of the functionality from an object code module does not form part of the final executable program but is linked to the executable program when the program is actually run. In other words, in a dynamic linking scenario, code from relevant libraries is loaded into the computer memory as needed. In general, static linking creates a derivative work, whereas dynamic linking may not under the GNU GPL.

The GNU GPL appears to offer a special exceptionfor software that is normally distributed with the operating system on which the application runs; namely, the licensee of a platform need not include, as part of the distribution of its application source code, any source code of the operating system. However, this does not seem to be much of an exception because the licensor of the operating system, and not the licensee, is the only one who can elect to publish the source code. Moreover, this exception assumes that it is the application that contains the open-source software and not the operating system. In the case of PaaS, the consumer needs to be concerned with the source code comprising the operating system.

By itself, the creation of a derivative work that is subject to the GNU GPL does not trigger the requirement that the source code for the whole program be released to the public. Rather, distribution of the derivative work triggers the obligation to release the source code for the whole program. The original GNU GPL did not account for distribution via a computer network, and providing access to the software through the web did not qualify as a traditional "distribution" of the derivative work. This situation was commonly referred to as the web distribution loophole. The FSF fixed the web distribution loophole by releasing the GNU Affero GPL, which defined distribution to include distribution through a network.

Conclusion

In the PaaS context, the end user needs to investigate whether the platform it is licensing contains any open-source software and which open-source software licenses govern the platform. If a copyleft license is applicable to the platform, then the end user needs to consider how it will link its proprietary application to the operating system. Although most applications dynamically link to the operating system, some developers may want the simplicity of static linking. If so, the developer needs to understand the potential issues with creating an executable with a static link to the operating system.

However, the issue for cloud-computing consumers will be the ability to get this level of detail from a cloud-computing provider, and even more challenging will be getting any representations in an agreement concerning the use of open-source software. Unless the amount of money involved is significant, most cloud-computing providers are unwilling to negotiate their contract terms with an end user. Regardless, if a cloud-computing consumer is to make an informed decision about whether to use a particular cloud-computing provider, it is important to understand the issues that may exist concerning open-source software.

Keywords: litigation, intellectual property, open-source software, cloud computing

Bradley J. Walz is a managing associate at Winthrop & Weinstine, P.A., in Minneapolis, Minnesota.

Complying with Source-Disclosure Obligations

By Edward J. Naughton – September 20, 2011

It may seem counterintuitive, but using open-source software code does not always require you to open up your source code. There are more than 50 approved open-source licenses, and many of them permit licensees to use or modify code without requiring the distribution of source code for the resulting work.

Copyleft licenses are a different story. Conceived by Richard Stallman, the founding figure of the free software movement, copyleft is a means of ensuring that everyone is free to use, copy, modify, and distribute software. Copyleft software is released under a license that allows downstream recipients to freely use, copy, modify, and redistribute the code, but it also requires any redistribution of the original code and derivations of it to fall under the same license. *See* Free Software Foundation, Inc., What Is Copyleft?, GNU Operating System.

Among copyleft licenses, the GNU General Public License (GPL) and Lesser General Public License (LGPL) are the archetypes, and they're some of the most popular. Stallman first wrote the GPL in 1989. His Free Software Foundation (FSF) released an updated version (GPLv2) in 1991, which became the most widely used free-software license. In 2007, the FSF released a new version of the GPL (GPLv3) that was designed to address some of the shortcomings of GPLv2, especially with respect to embedded systems. (The obligations created by GPLv2 and GPLv3 are substantially identical in many respects, and I refer to them collectively as “the GPL” when there is no need to distinguish between them.)

The LGPL (originally the Library General Public License) was developed by the FSF as a less-copyleft alternative to the GPL for use with libraries. It permits a non-copyleft application to use free software libraries in certain circumstances without making the application itself subject to copyleft. It requires, however, that the library and any modifications to it remain under the original copyleft license. Version 2.1 (LGPLv2.1), released in 1999, remains the most common version, although version 3 (LGPLv3) was released in 2007. (Again, I’ll refer to these versions collectively as “the LGPL” when there is no need to distinguish them.)

Lawyers who work with software companies know that it can be a challenge to determine whether the GPL or LGPL requires the distribution of source code, but the challenges don’t end there. Simply complying with the GPL can be tricky, too, particularly when the code is embedded in a device. The consequences of a mistake can be serious. In fact, all of the recent GPL-enforcement cases have turned not on whether the GPL applied, but on whether the defendant had complied with the source-code-disclosure requirements.

What Must Be Disclosed?

GPLv2

GPLv2 requires the provision of a copy of the license and the complete “corresponding source code” of works subject to the license. “Source code” is “the preferred form of the work for making modifications to it.” GPLv2, § 3. For an executable, source code “means all the source code for all modules it contains.” But it also includes “any associated interface definition files” plus “the scripts used to control compilation and installation of the executable.” Because the purpose of the license is to permit a user to modify the original code and generate a new executable, “scripts” includes any software programs, tools, or scripts necessary for a user to install a modified version of the program, such as makefiles, configuration files, build scripts, and packaging scripts.

It usually is not necessary to provide the compiler used to generate the executable, but you must provide information about it. The Software Freedom Law Center (SFLC), one of the primary enforcers of the GPL, recommends that you provide a README file with the name of the compiler, its exact version number, and where it can be acquired, regardless of whether the compiler is proprietary or open source. A Practical Guide to GPL Compliance, Software Freedom Law Center. When the GPLed code is embedded in a device, however, it is usually necessary to provide the cross-compiler and toolchain so a user can build the modified software and reinstall it in the device. A detailed README file should describe, step by step, how to build a binary from the sources.

Because GPLv2 requires the provision of “corresponding” source code, the source code must correspond exactly with the executable that is distributed. Frequently Asked Questions about the GNU Licenses, GNU Operating System. If multiple versions of firmware for a product have been distributed, then the source code corresponding to each such version must be distributed. Good version control, release-management practices, and build documentation are critically important if you want to avoid problems later.

GPLv3
The source-code-disclosure requirements for GPLv3 are similar. "Corresponding Source" means "all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities." GPLv3, § 1. Thus, like GPLv2, GPLv3 requires the provision of all of the programs, files, tools, and scripts necessary to install a modified version of the GPLv3 software, including the compiler or information about it. A copy of the license must also be included.

In addition, when GPLv3 software is distributed in a "User Product," such as a consumer device, it also is necessary to provide "Installation Information." Installation information includes any methods, authorization keys, or other information required to install and execute modified versions of the GPLed code in that user product. The only time installation information does not need to be provided is if the GPLv3 software cannot be modified as a technical matter (for example, if the software is burned onto a ROM chip that cannot be upgraded without replacing the chip), as opposed to a contractual provision that purports to prohibit modification.

LGPLv2
The source-code-disclosure requirements for the LGPL parallel those of the GPL. LGPLv2 requires the disclosure of the license and complete corresponding source code. "Source code" is the preferred form of a work for making modifications to it. LGPLv2.1 § 0. For a library, "complete source code" means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library. LGPLv2 similarly requires the provision of all of the programs, tools, and scripts necessary to install a modified version of the library, and there must be a 1:1 correspondence between the source and the executable.

LGPLv3
The source-code-disclosure requirements for LGPLv3 depend on how an application interacts with the licensed library. If the application merely incorporates material from the library's header files, it's not necessary to provide source code; it is enough to include copies of the LGPLv3 and GPLv3 licenses and a prominent notice that the library is used by the application and is covered by the LGPLv3. § 3.

If the application is combined with or linked to the library to form a combined work, the source-code-disclosure obligations are more complicated. In addition to including the information described above, it is also necessary to provide the library in a way that permits the user to modify the LGPLed library and recombine or relink it with the application. There are two ways to do this. The first is to provide the "Minimal Corresponding Source" of the combined work, which is the corresponding source code for the combined work, excluding source code for portions of the combined work that are based on the application and not on the library. *Id.*, at § 4(d)(0).

In addition, the corresponding application code must be provided. The corresponding application code may be provided in object code or source code form, along with all data and utilities needed to reproduce the combined work, so long as it is provided in a form suitable for and under terms that permit a user to recombine or relink the application with a modified version of the LGPLed library. Alternatively, it is sufficient to provide a shared-library mechanism that uses at runtime a copy of the LGPLed library already on the computer system, as long as it will operate properly with a modified version of the library. *Id.*, at § 4(d)(1). Finally, it is necessary to provide installation information to the same extent as it is required under GPLv3 (in other words, if the L GPLed code is in a user product that can be field-upgraded). *Id.*, at § 4(e).

How Must the Corresponding Source Code Be Provided?

GPLv2 and LGPLv2

There are three options for providing corresponding source code under GPLv2 and LGPLv2. The corresponding source code can be provided alongside the executable, GPLv2, § 3(a), or the executable can be accompanied by a written offer, valid for at least three years, to give any party the corresponding source code. *Id.*, at § 3(b). Unfortunately, the license, reflecting the prevailing technology when it was drafted, requires the source code to be provided on “a medium customarily used for software interchange.” In other words, the code must be on physical media; Internet distribution is not sufficient under GPLv2. Therefore, while it is very common for the written offer (usually in the user documentation) to include a link to a download site, that is technically insufficient.

Section 3(c) of GPLv2 permits a licensee to pass along a section 3(b) written offer, but that is expressly limited to “noncommercial distributions” and only if the licensee received the executable with such a written offer.

GPLv3 and LGPLv3

GPLv3 permits source-code distribution by the same methods as GPLv2, but it also allows some additional options. First, if you offer the object code for download over the Internet, you can offer the corresponding source code over the Internet as well, in the same way and from the same place as the object code. GPLv3, § 6(d). Similarly, if the object code is offered over a peer-to-peer network, the source code can be offered in the same way. In addition, GPLv3 fills the gap in GPLv2; the written offer to provide source codes can be fulfilled by Internet distribution if the download site for the source codes is publicly available and remains active for three years from the last distribution of the binaries (or spare parts for a device containing the binaries). GPLv3, § 6(b). LGPLv3 authorizes distribution by the same methods as GPLv3 or by means of the shared library mechanism described above.

As a practical matter, it may not be possible to take advantage of the new and more convenient methods of source code distribution allowed by GPLv3. Software distributed commercially is almost always subject to multiple licenses, and it is quite likely that a device that includes GPLv3 code also includes code subject to GPLv2 and/or LGPLv2. In that event, the only permissible methods of source-code distribution will be the more restricted methods authorized by GPLv2.

Who Has the Obligation to Provide Corresponding Source Code?
Everyone who distributes (GPLv2/LGPLv2) or conveys (GPLv3/LGPLv3) code must provide source code. In the context of a consumer device, the source-code-provision obligation attaches to every level of the distribution chain, although the scope of those obligations will vary at each stage. Consider, for instance, a DVD player that is sold by an electronics retailer and built by a contract manufacturer, which incorporates a chip obtained from a chip manufacturer whose chip has firmware containing an embedded program subject to GPLv2 that was written by an independent software developer. The original developer makes the source codes for his or her program available under the GPLv2. The chip manufacturer can provide the corresponding source code alongside the binaries on the physical medium of the chip, but will not typically take up that precious space and will instead make a written offer under GPLv2 § 3(b) to provide source code to anyone who asks. Because the contract manufacturer is engaged in commercial distribution, it cannot simply pass along the chip manufacturer's written offer and must make its own source-code provision. The retailer, in turn, must do the same.

This is no mere hypothetical. GPL-enforcement actions have been asserted against retailers such as Best Buy; distributors such as Verizon; device manufacturers such as Cisco Systems, Monsoon Multimedia, JVC, and Samsung; and chip manufacturers such as Broadcom. *See*, *e.g.*, *Software Freedom Conservancy, Inc. v. Best Buy Co.*, 09-CV-10155 (S.D.N.Y.) (naming 14 defendants); *Andersen v. Monsoon Multimedia, Inc.*, 07-CV-8205 (S.D.N.Y.); *Andersen v. Verizon Communications, Inc.* 07-cv-11070 (S.D.N.Y.). Companies at every level of the distribution chain need to be mindful of GPL source-code-disclosure obligations.

When Must the Corresponding Source Code Be Provided?
As discussed above, the preferred method for providing corresponding source code is alongside the executable, on the same media, on a CD accompanying a device, or on the same download site. The timing issue arises only when the corresponding source code does not accompany the binaries, when the licensee gives a written offer to provide the source code. The text of the GPL does not expressly specify timing, but the structure and purpose of the license make plain that the corresponding source code must be available simultaneously with the distribution of the object code. The enforcers of the GPL emphasize this:

> It is unacceptable to use option [3](b) merely because you do not have Corresponding Source ready. We find that some companies chose this option because writing an offer is easy, but producing a source distribution as an afterthought to a hasty development process is difficult. The offer for source does not exist as a stop-gap solution for companies rushing to market with an out-of-compliance product. If you ship an offer for

> source with your product but cannot actually deliver *immediately* on that offer when your customers receive it, you should expect an enforcement action.

A Practical Guide to GPL Compliance, Software Freedom Law Center (emphasis in original); *accord* gpl-violations.org Source Code Release FAQ, gpl-violations.org ("The GNU GPL demands that as soon as you distribute GPL licensed software in executable format you make available the 'complete corresponding source code'."); *but see* Google to Delay Open Source Honeycomb Release, Linux for Devices (describing indefinite delay in release of Android Honeycomb source code).

Companies distributing GPLed code must therefore plan ahead to ensure that they are able to provide corresponding source code simultaneously with the shipment of their executables and devices.

What Are the Consequences of Noncompliance?
For many years, the FSF enforced the GPL and LGPL through private negotiations rather than litigation, and there were questions about whether the GPL was enforceable. *See* Eben Moglen, "Enforcing the GPL," (2001). Perhaps because of these doubts, starting in 2007 the FSF and its allies, the SFLC and the Free Software Conservancy (FSC), began more aggressive enforcement efforts, including filing copyright-infringement actions. Those lawsuits give greater visibility into the enforcement process.

The heart of those enforcement actions is section 4 of the GPLv2 (section 8 of the LGPLv2.1), which provides that any violation, even if inadvertent, results in the automatic and immediate termination of rights to use the licensed code. Curing the violation does not automatically reinstate the license; it is necessary to obtain an explicit reinstatement from the copyright holder:

> Since your rights under GPLv2 terminate automatically upon your initial violation, all subsequent distributions are violations and infringements of copyright. Therefore, even if you resolve a violation on your own, you must still seek a reinstatement of rights from the copyright holders whose licenses you violated, lest you remain liable for infringement for even compliant distributions made subsequent to the initial violation.

A Practical Guide to GPL Compliance, Software Freedom Law Center.

As a legal matter, this position has fair support. In fact, in 2007, a German court relied on that very provision to impose an injunction in a case involving Harald Welte, the founder of gplviolations.org and a well-known GPL enforcer in the EU. *See* Landgericht [LG] München I [Lower Regional Court of Munich], *Welte v. Skype Technologies, SA* [PDF], File No. 7 O 5245/07 (July 12, 2007); *see also* Skype Hangs Up on Appeal, Will Fully Comply with GPL, Ars Technica. Recent U.S. cases similarly support the proposition that the breach of a copyright license can result in the loss of rights under the license, such that the continued use or distribution of the work is infringing and subject to an injunction. *See, e.g.*, *MDY Industries v.*

Blizzard Entertainment, 629 F.3d 928 (9th Cir. 2010) (breach is infringement if the licensee breaches condition on a grant of license and the condition is related to exclusive rights under copyright); *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008) (breach of condition on grant of a license is copyright infringement); *LGS Architects v. Concordia Homes*, 434 F.3d 1150 (9th Cir. 2006) (entering an injunction where the licensee exceeded the scope of the license).

The provision of corresponding source code will always be required before reinstatement, but the copyright holders usually demand additional concessions. Many of the reported settlements involve the appointment of an "open source compliance officer," periodic reporting of compliance efforts, notification to previous recipients of the GPLed program, and monetary payments. *E.g.*, BusyBox Developers and Monsoon Multimedia Agree to Dismiss GPL Lawsuit, Software Freedom Law Center (Monsoon Multimedia settlement); FSF Settles Suit Against Cisco Free Software Foundation (Cisco settlement); BusyBox Developers Agree to End GPL Lawsuit Against Verizon, Software Freedom Law Center (Verizon and ActionTec settlement).

The SFLC takes the view that copyright holders have broad latitude in their reinstatement demands. A Practical Guide to GPL Compliance, Software Freedom Law Center ("Different copyright holders condition reinstatement upon different requirements, and these requirements can be (and often are) wholly independent of the GPL. The terms of your reinstatement will depend upon what you negotiate with the copyright holder of the GPLed program."). And they do make significant demands: According to affidavits filed in a case brought by the developers of Busybox against Best Buy, the SFC demanded that Broadcom (Best Buy's supplier) disclose the source code of a number of proprietary libraries and programs as a condition to reinstatement. When Broadcom refused, the SFC sought an injunction against Best Buy's sale of a line of Blu-ray players. Declaration of Rashid Khan, Doc. No. 180, in *Software Freedom Conservancy, Inc. v. Best Buy Co.*, 09-CV-10155 (S.D.N.Y.). According to other filings in that case, the FSC also demanded the right to review, prerelease, new versions of products and to review code not authored by the developers it represents.

The GPLv3 and LGPLv3 make it easier to resolve violations. Under those licenses, a licensee's rights are restored upon curing the noncompliance, unless a copyright holder provides notice of violation within 60 days. If the licensee has not previously been notified of a violation by that copyright holder and it cures the violation within 30 days of such a notice, its rights are restored. Otherwise, it must negotiate with the copyright holder for explicit reinstatement of the license, as under GPLv2.

Noncompliance can lead to monetary damages as well. Most reported settlements have involved some monetary payments, and, in the *Best Buy* [PDF] case, the plaintiffs obtained statutory damages of $90,000 and attorney fees from Westinghouse after it ceased defending the case upon its insolvency.

Conclusion
The GNU licenses are complicated in most respects, and the source-code-provision requirements

are no exception. The challenges they present are not going away, but are instead growing more important. As free software has become more widely accepted and as software-supply lines have grown longer and more convoluted, the potential for inadvertent violations increases. At the same time, GPL enforcement has become more active, making the repercussions of noncompliance more real and much more significant. Any lawyer who advises software clients must understand the GNU licenses and educate his or her clients to ensure they are not unwittingly setting themselves up for an embarrassing and costly compliance action.

Keywords: litigation, intellectual property, GNU, copyleft, GPL, LGPL, source code

Edward J. Naughton is a partner at Brown Rudnick, LLP, in Boston, Massachusetts.

The Economic Incentives of Open-Source Software

By Jennifer Vanderhart – September 20, 2011

Why would someone put time and effort into making computer software and then give it away for free? Most of us can probably think of a few reasons, and many of them would apply to any good or service. Today, there are many companies and people who provide access to computer software and receive no money for that access. Examples include freeware and software licensed as open source.

The number of projects that use open-source software has grown exponentially. But to understand how open sourcing developed into a significant business model for providing access to software, we need to explore the relatively recent beginnings of the software industry and the economic nature of software as non-rivalrous and generally non-excludable. The various economic incentives motivating initial and continuing contributions to open-source software, as well as the increasing usage of open-source software options, have allowed for various business models to develop based on what is essentially something that can be obtained for free.

Recent, albeit limited, court decisions, as well as settlements in litigation brought by open-source plaintiffs, have recognized and reinforced certain incentives to contribute to and use open-source software, while the outcome of other cases might have the effect of limiting usage to some extent. Other judicial decisions, while tangential to the actual provision of open-source software, may also affect the provision and licensing of open-source software.

In the Beginning

Economists have examined a number of motivating factors behind the provision of open-source software. In the early 1960s, much of the writing of code was done by either academics in a university setting or researchers in a corporate environment, AT&T in particular. These programmers, or “hackers” as they were sometimes known, often shared ideas with each other. It was the source code that was shared—the actual text written by a computer programmer in a computer programming language. Source code is distinguished from object code, which is the binary language that the computer actually uses. It is the difference between words and equations readable by humans and a series of zeros and ones that can generally only be interpreted by a computer. It is much more difficult, if not impossible, to modify a program if the source code is not made available. Most proprietary programs sold to end users provide only the object code and, moreover, require the user to sign a license that prohibits them from attempting to modify, reverse-engineer, or otherwise tamper with that object code.

Collaboration among different companies and academics became increasingly strained as investments in, and profits from, software increased. However, the movement to open source as it is known today was aided by two events in the early 1980s. The first event was the passage of a 1980 amendment to the Copyright Act that allowed for copyright protection to be extended to object code. Prior to the amendment, software could be protected under copyright; but for that to happen, the source code had to be published and registered. This eliminated any trade-secret protection that the software might have enjoyed up to that point, and the would-be copyright holder did not always view this as favorable. After the amendment, the software owner could obtain a copyright on the object code while simultaneously maintaining the source code as a trade secret.

The second event resulted from an antitrust case brought by the U.S. government against AT&T. The case was settled in 1982 and, as part of the settlement, a restriction that had prevented AT&T from selling software was lifted. Shortly thereafter, AT&T began to threaten litigation over intellectual property rights it believed it owned related to the Unix software platform. A number of researchers from various universities had contributed improvements and additions to the Unix software over time. Although AT&T had been collecting a licensing fee for its contribution, it was a nominal amount. The software had developed into a full operating system, the benefits of which AT&T threatened to capture fully with increasingly higher Unix license fees.

One response to AT&T's actions came from Richard Stallman, of the Artificial Intelligence Lab at the Massachusetts Institute of Technology (MIT), who announced in 1983 that he was going to write a free version of Unix. He started the GNU Project, which eventually developed in the Free Software Foundation (FSF). The very name of the project illustrated the tension between those who wanted to commercialize the software and those who wanted to keep it unfettered—GNU was a recursive acronym (of which there are a number of computer-related contemporaneous examples) that stood for "GNU's Not Unix." One of the most significant contributions the FSF made to the open-source community was the development of the GNU General Public License (GPL), still one of the most widely used open-source software licenses today. There are many other open-source software licenses, some more and some less restrictive in their requirements, as well as some that are considered special-purpose licenses, but for the most part, they are largely based on the GPL's definition of free software. These include Mozilla, Apache, the MIT license, the GNU Library (or Lesser) GPL, the Educational Community License, and the IPA Font License.

From an economic perspective, it is important to understand what exactly was meant by "free" in the open-source licensing context and how that concept was understood when the GPL was implemented. Stallman suggested that "you should think of 'free' as in 'free speech,' not as in 'free beer,'" and he defined free software as that which allowed the following four freedoms (which he counted starting at zero):

0. The freedom to run the program for any purpose.
1. The freedom to study how the program works and change it to make it do what you wish.
2. The freedom to redistribute copies so you can help your neighbor.
3. The freedom to improve the program and release your improvements (and modified versions in general) to the public so that the whole community benefits.

Free, then, was never intended to necessarily mean "no charge," although realistically, the price of the software itself will generally be driven down to zero for anyone willing to accept the other terms of the GPL license. This is because the four freedoms include the freedom to redistribute copies, and because software is non-rivalrous, meaning one person's benefit is not reduced when another person also uses the software (and the benefit may even be increased), the ease of communication over the Internet reduces transmission costs to virtually zero, such that software code can easily and perfectly be shared. Nonetheless, not only have programmers found it to be in their best interest to contribute to open-source software, but entire companies have been founded on the basis of a product that can essentially be obtained for free.

Economic Incentives Behind Open-Source Usage and Contributions

Clearly, not everyone who uses open-source software contributes to its code. For instance, just because you can download and use the open-source Firefox web browser for free does not mean you have either the skills or the inclination to modify it or write your own add-ons. Nonetheless, many people do devote their time and energy to modifying open-source software, creating their own add-ons and more. Unless one is to assume that these contributors, and others providing code for other open-source software, are acting in an economically irrational manner, these contributors must be deriving positive utility from their contributions.

Especially in the early days of open-source software, many of the programmers contributed voluntarily, motivated by the desire to learn and the satisfaction of having a community of like-minded individuals able to appreciate their contributions. Some also hoped for a reputational effect—that prospective employers might be impressed with their skills. One of the requirements of releasing a modified version of open-source software licensed under the GPL is that the copyright holder writes a copyright notice in his or her own name. The terms of the GPL quite literally require individuals to be recognized as the source of their code. So, while programmers might not have been selling open-source software or services, they were often essentially selling themselves through their contributions to the software.

Different incentives motivate companies such as Red Hat and Novell, which are based on providing consulting, training, and other services for Linux-based systems. Linux is an open-source operating system, referred to as “Unix-like,” and it is currently used widely. Red Hat in particular, the largest company providing these services, uses the fact of its continuing contributions to the Linux open-source code as a selling point. According to its 2011 annual report, Red Hat earned more than $900 million in revenue in fiscal year 2011, 85 percent of which was from subscriptions to its support services, and it had a Generally Accepted Accounting Principle (GAAP) operating margin of 16 percent. Red Hat, Novell, and other like-minded companies have also expanded into the business of providing proprietary extensions for the core open-source Linux system. Companies that base their business model on providing support for open-source platforms are also motivated to contribute to open-source software, but they may have different incentives than the voluntary individual contributors.

A somewhat more recent development is dual licensing, whereby a copyright owner will distribute the same software under either a proprietary license or under an open-source license, typically the GPL. The open-source licensee may not pay a fee, but the license comes with other requirements, as discussed previously—in particular the requirement to disclose any changes or additions to the software. This may not be a feasible option for certain businesses wanting to make changes that they would like to keep out of the public eye. On the other hand, there are many businesses that simply want to use the software as it exists or that do not derive any, or enough, value from keeping their modifications from the open-source community. Through this product differentiation, a company can hope to increase its consumer base as well as its profits.

Another reason for the use of, and contribution to, open-source software has been referred to as the “hold-up” problem. If a user requires a modification after the initial licensing of a proprietary software product, the user is at least partially at the mercy of the software provider. The licensee may have invested time and effort (and, therefore, “dollars”) into a given software package and realizes that there would be costs to transitioning to a different software package. The software provider is aware of this as well and would likely take it into account when setting a price for modifications and updates, especially for a customized modification. Because it is not possible to contract for all eventualities at the time of the initial license, this might reduce the licensee’s incentives to make complementary investments or purchase the software in the first place. The larger the complementary investments, the less likely it is that a company would want to use an option from a different software provider. Using an open-source option avoids the hold-up problem.

End-user companies and individuals contribute to and use open-source software to have software that fits their particular needs, that can be customized, and that can be obtained at a lower cost than a proprietary option. By some estimates, it is this type of contributor who has had the largest role in the advancement of open-source software. Because software is often not the business of these companies but rather a means to an end, they are less likely to feel proprietary about the contributions they make.

One of the costs of using open-source software includes the cost of complying with the license requirements, particularly the requirement to provide or offer to provide the source code. For the most part, this is not an overly burdensome requirement but it is fundamental to the advancement of open-source software.

Litigation

The litigation related to open-source software can be placed into two general categories. The first category comprises copyright suits brought by those in the open-source community against companies that violate the terms of the GPL, such as *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008), and the *BusyBox* cases. The second category comprises intellectual property infringement matters, for the most part, patent-infringement cases filed against companies in the open-source community. The outcomes could have economic effects on the broader open-source community.

Jacobsen v. Katzer *and the* Busybox *Cases*

In *Jacobsen v. Katzer*, 535 F.3d 1373 (Fed. Cir. 2008), the court acknowledged what are often the non-pecuniary economic benefits to those who contribute to open-source software, stating that there "are substantial benefits, including economic benefits, to the creation and distribution of copyrighted works under public licenses that range far beyond traditional license royalties." *Id.* at 1379. The court gave examples of specific benefits (a subset of those discussed above), including the generation of market share and reputational effects, and went on to state that the "choice to exact consideration in the form of compliance with the open source requirements of disclosure and explanation of changes, rather than as a dollar-denominated fee, is entitled to no less legal recognition." *Id.* at 1382.

As discussed, many of the incentives underlying the provision of, and contributions to, open-source software are based on the restrictions contained in licenses such as the GPL. The court recognized the importance of not destroying those incentives and concluded as follows:

> The clear language of the Artistic License creates conditions to protect the economic rights at issue in the granting of a public license. . . . The attribution and modification transparency requirements directly serve to drive traffic to the open source incubation page and to inform downstream users of the project, which is a significant economic goal of the copyright holder that the law will enforce.

Id.

A number of recent lawsuits filed by the Software Freedom Conservancy (SFC) involve similar issues. The SFC was established by the Software Freedom Law Center and has filed claims against at least 20 companies since 2007 alleging violations of the terms of the GPL used to license BusyBox, a software application based on the Linux kernel. The most recent case, brought against 14 companies, alleges that the defendants have infringed the plaintiffs' copyrights by not providing "either (i) the 'complete corresponding machine-readable source code' or (ii) a 'written offer . . . to give any third party . . . a complete machine-readable copy of the corresponding source code.'" The defendants include Best Buy, Samsung, Westinghouse, and JVC, and the products that are accused of containing the software include high-definition televisions, digital video recorders (DVRs), DVD players, video cameras, and wireless routers. The plaintiffs are seeking a permanent injunction, as well as actual damages and any additional profits or, alternatively, statutory damages.

The SFC has settled a number of the cases it has filed for undisclosed settlement amounts and compliance with the terms of the GPL. One of the defendants, Westinghouse Digital Technologies, which had applied for bankruptcy in California, refused to participate in discovery. Consequently, the court granted SFC triple damages of $90,000 for willful copyright infringement, almost $50,000 in legal fees, and an injunction against Westinghouse. In addition, the judge ordered all infringing high-definition televisions to be forfeited to the plaintiff.

The decision in *Katzer*, as well as the opinion and order against Westinghouse, should provide incentives to companies that license open-source software to comply with the terms of the GPL. Licensees of other open-source licenses may also be encouraged to look carefully at their policies. In addition, these developments will serve to enforce the incentives of many who already contribute to, and those who may be thinking of contributing to, open-source software.

Lawsuits Against Red Hat and Google
A number of patent-infringement lawsuits have been filed against Red Hat, as well as other companies that use open-source software in their products, although Red Hat appears to be the biggest target. Not surprisingly, Red Hat submitted an amicus brief to the U.S. Supreme Court asking it to affirm the *Bilski* decision limiting the patentability of business methods, which, in Red Hat's opinion, included software. The opinion of the Court in *Bilski* did not place software outside the protection of the patent system.

Red Hat's record in court is mixed. It successfully defended a lawsuit filed by IP Innovation, a unit of Acacia. The case was in the Eastern District of Texas, typically considered a plaintiff-friendly venue for patent litigation. Nonetheless, the jury not only found that Red Hat did not infringe but also invalidated the patents at issue. On the other hand, Red Hat's settlement of another patent-infringement case filed by Firestar may not have been quite as satisfying. However, as part of the settlement, Red Hat negotiated that both licensors and licensees of the Red Hat product would be covered. At least for the time being, this should help to maintain incentives for those companies to continue to contribute to and use open-source software.

In April of this year, a jury ruled against Google in a patent-infringement suit filed by Bedrock Computer Technologies. This case was also filed in the Eastern District of Texas. Bedrock has also sued MySpace, Paypal, Yahoo, Amazon, Match.com, AOL, and two small Texas companies. In all likelihood, the two small Texas companies were included so the plaintiffs could get their venue of choice. While $5 million is not a large damages amount for a company like Google, the patent covers certain functionality of the Linux kernel, and after Google's loss, other companies have decided to enter into licenses with Bedrock. Of course, if there are enough patentees who can claim infringement and demand licenses for Linux-based, open-source software, the economic viability of the many systems based on Linux will be brought into question.

One of the most significant matters related to open-source software, at least in terms of potential damages, is working its way through the courts now. In *Oracle v. Google,* currently slated for trial in November 2011 in the Northern District of California, Oracle was seeking between $1.4 and $6.1 billion dollars in a patent-infringement case involving the Android mobile operating technology. Google requested leave to file a *Daubert* motion to exclude the testimony of the damages expert, who used an entire market-value-rule approach to the royalty base and a royalty rate of 50 percent. The court recently struck the expert’s testimony as unreliable, with leave to try again and guidance for testimony that would be admissible. Perhaps because the actual damages amounts were released so recently or perhaps because it is Google, which, even compared with Red Hat, is just so darn rich and big, but the reaction of the open-source community has not yet been the vast outrage that one might have expected.

Conclusion

The incentives behind the use of and contributions to open-source software vary widely. Certain terms in open-source licenses are important to maintaining those incentives and have been upheld as valid by the courts. Other litigation affecting the open-source community in general and patent-infringement lawsuits in particular may challenge the viability and growth of open-source software.

Keywords: litigation, intellectual property, open-source software, incentives

Jennifer Vanderhart is a principal economist at Exponent in its Alexandria, Virginia, office.