

OREGON INTELLECTUAL PROPERTY NEWSLETTER



Oregon State Bar
Intellectual Property Section

Intellectual Property Student Organization
Lewis and Clark Law School



Volume 6 • Number 2

Published by the Intellectual Property Law Section of the Oregon State Bar

Winter 2006

Inside this Issue

- 1 Open Source In Practice:
From Idea To Market To
Your Desk
- 6 Bringing FOSS Into The
Enterprise: Thoughts On
“Best Practices”
- 11 Open Source in M&A and
Other Transactions
- 15 Implied Patent Rights and
Open Source Software
- 19 Harald Welte, Linux, and
the GPL
- 22 Semi-Open Source: The
Case for Hybrid Licensing
- 27 Trademark Enforcement
for National Brands
Made Easier – Not So
For Others: Significant
Changes to Trademark
Dilution Act Signed
into Law

Open Source In Practice: From Idea To Market To Your Desk



By Anne Glazer

Shareholder, Lane Powell P.C.

Outgoing Intellectual Property Section Chair

GlazerA@LanePowell.com

Are we on the cusp of a paradigm shift? Are developers and proponents of open technology changing the world, or at least the worlds of software and entertainment? Is “open source” just the latest fad, like “dot-com” before it? Who cares, if you can’t corner the market for a software product because you can’t control access to it? And what does any of this have to do with my law practice?

Whether you are an enthusiast or a skeptic about free and open source software and technology, this newsletter issue should be useful in your technology practice. At the most basic level, “open source” is a type of software license you need to know about. (Terminology is a bear in this fluid world – I will use “open source” to mean free, libre or open source software.) Open source software is changing the way some people do business, while for others it is simply another legal issue. In any case, open source issues arise consistently in the practice of law.

At the cosmic level, the open technology movement has potential to change more than just software. As *The Economist* wrote:

Though the term [open source] at first described a model of software development (where the underlying programming code is open to inspection, modification and redistribution), the approach has moved far beyond its origins. From legal research to biotechnology, open-business practices have emerged as a mainstream way for collaboration to happen online.¹

Free Software Foundation counsel Eben Moglen pointed out, in “Anarchism Triumphant,” that a bitstream is a bitstream. As everything moves from analog to digital representation, all forms of human symbolic activity become “software.”² Music, television, news, and other forms of information – all reduce to ones and zeroes. When you

continued on page 2

“If you’re like me, you find these ideas entertaining and important. But let’s face it: unless you’re Lawrence Lessig, they are unlikely to pay the mortgage.”

think about it, all our perceptions may be converted to code. Indeed, some scientists conceive of the universe itself as a computer program. Hence the epic policy battles between “intellectual property” forces (e.g., Microsoft, Hollywood, the recording industry, the pharmaceutical industry) and those who oppose them (e.g., software developers, file-sharers, hippies, librarians, developing-world governments, and at least half of Silicon Valley). The “IP rights” side represents powerful economic interests built around a paradigm that has developed, changed, and worked pretty well over the centuries. “The Congress shall have Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries . . .”³ The “anti-IP” side arguably represents the developing forces of technology itself.

Digital technology puts increasing pressure on the established paradigm. “Information wants to be free.” This aphorism is generally credited to that old hippie and early Internet adopter, Stewart Brand. The debate is far from new. Indeed, the full statement, more than twenty years old, is more nuanced:

Information Wants To Be Free. Information also wants to be expensive. Information wants to be free because it has become so cheap to distribute, copy, and recombine – too cheap to meter. It wants to be expensive because it can be immeasurably valuable to the recipient. That tension will not go away. It leads to endless wrenching debate about price, copyright, ‘intellectual property,’ the moral rightness of casual distribution, because each round of new devices makes the tension worse, not better.⁴

If you’re like me, you find these ideas entertaining and important. But let’s face it: unless you’re Lawrence Lessig, they are unlikely to pay the mortgage.

So, where is the money? Debates rage about whether open source business models can be profitable in the long run. In these early days, there are encouraging signs for those in favor of those models. Most importantly,

Linux is the world’s fastest-growing operating system. According to an analysis prepared for Open Source Development Labs, overall Linux marketplace revenues for servers, PC hardware, and packaged software are expected to reach \$35.7 billion by 2008, at an annual growth rate of 26%. Leading Linux provider Red Hat is a Fortune 100 Fastest Growing Company, based on profit and sales growth.⁵

The red-hot question is whether open source business models can be widely profitable for software enterprises beyond the operating system. Open source software is essentially code that is developed and licensed in “developer-favorable” ways. How important are developers? Can an “open source” company enjoy the levels of competitive advantage and barriers to entry that “closed source” IP has provided? These questions are being tested right here in Oregon by various enterprises, from startups to Fortune 500 companies.

First,⁶ many technology companies that are not built around open source nevertheless profit by incorporating it into their businesses. For example, companies from Intel to Oracle are optimizing their products to work well with Linux. In a related development, many of these companies are significant patrons of open source developer communities. For example, IBM’s massive support of Linux, Apache and Eclipse helps it protect its competitive position and creates many diverse opportunities for IBM to sell its products and services.

Device companies are also choosing to embed open source software in their products. TIVO is the most notable example. Proprietary code is not their profit opportunity; instead, they take advantage of the low cost and other advantages of open source development for their software and firmware.

Second, companies that develop open source software have found profitable ways of going to market, even though they cannot control the distribution of their open source code. A few different market options are commonly considered:

Mixed IP: The enterprise may offer some modules under an open source license, most likely because they are modifications or enhancements of open source soft-

ware. However, it may have other developments that it chooses to license for a fee. The company's overall product to the customer is the assemblage of all the pieces.

Dual licensing: The enterprise offers users a choice: (a) an open source license to the software, perhaps on a trial basis or (b) "commercial grade" software for a fee. There are any number of reasons why a user may choose to pay for software it could get for free. The commercial version may come with commercial distribution rights, a larger set of features, warranties and indemnification against infringement, or other valuable add-ons.

A leading practitioner of this model is MySQL, a leading database provider that is estimated to have more database installations than IBM or Oracle. MySQL reportedly has \$40M in revenue in the past twelve months, and by one estimate the company may be valued at over \$1 billion. MySQL's commercial version comes with access to support services, a knowledge base, and certification (i.e., warranties).

Services/consulting: In general, more than 70% of what enterprises spend on software goes for services. Some companies that have tried to "go big" on a standalone services model for open source software have foundered because they lacked a protected position. However, those that combine it with their own developments seem to be profiting from it. MySQL is one of these. Red Hat (Linux), its acquisition JBoss (middleware), and Compiere (ERP) are other prominent examples. Maintenance services may be offered on a subscription basis, just as with any software.

Application Service Provider (ASP): The enterprise uses open source software to offer web-based services. Examples include CRM services such as those hosted by Salesforce.com and NetSuite.

Venture capital investment in open source startups was all the buzz in 2005. Although the buzz seems to have worn off a bit, the interest is still there. Compiere, an open source company that moved to Portland in 2005, landed \$6 million in venture capital just a few months ago.

The sweet taste of local success was short-lived, however, as Compiere took its \$6 million and immediately moved to Silicon Valley to be closer to its venture backers and a larger pool of potential employees.

So, the local story is mixed. None of the companies mentioned in this article so far is headquartered here. However, a solid case can be made that Oregon has

continued on page 4

OREGON INTELLECTUAL PROPERTY NEWSLETTER

Staff, Winter 2006

Executive Editors

Duke Tufty dukedt@gmail.com
Editor-in-Chief

Sean Walsh walsh@lclark.edu
Submissions Editor

Form & Style Editors

Michael Massa mamassa@lclark.edu
Greg Touchton touchton@lclark.edu
Elizabeth Woodward ewoodard@lclark.edu
Peter Tovey ptovey@lclark.edu

Layout Editors

Jesse Abrams abrams@lclark.edu
Megan Murray meganaimee@gmail.com

Text Editors

Ann Trader atrader@lclark.edu
Yoonhee Chang ychang@lclark.edu
Sarah Petersen sarahp@lclark.edu
Michael Nelson nelson@lclark.edu
Wendy Packard dubyaprhit@yahoo.com
Deneil Bragg dbragg@lclark.edu
Kerry Snyder snyder@lclark.edu
Jen Coatesco coatesco@lclark.edu
Bryan Beel beel@lclark.edu
Stephen James sjames@lclark.edu
Anya Ronshaugen asuch@lclark.edu
Joseph Pitt jpitt@lclark.edu
Milos Bosanac Milos@lclark.edu

Faculty Advisor

Professor Joseph S. Miller
jsmiller@lclark.edu

This newsletter is available online:
<http://www.lclark.edu/~ipso/OIPN>

Visit the OSB IP Section website:
<http://www.osbar.org/sections/ip.html>

Submissions

Sent to: oipnews@lclark.edu

The Newsletter follows the conventions of the Association of Legal Writing Directors & Darby Dickerson, ALWD Citation Manual (Aspen L. & Bus. 2000).

a unique opportunity with the open source experiment. Oregon is home to many of the leading companies, institutions, and executives driving the global market for open source software and hardware. The local area boasts:

Open Source Development Labs: Sponsor of Linux and its creator Linus Torvalds. <http://www.osdl.org>. OSDLs sponsors and constituency are global.

Oregon State University Open Source Lab: Home of the Linux kernel and many other major open source projects worldwide. <http://www.osuosl.org>.

Open Technology Business Center: A growing business incubator supported in part by the City of Beaverton. There are currently seven startups in residence. <http://www.opentechcenter.com>. (However, OTBC has moved away from an “open technology only” model to welcome all kinds of startups.)

O’Reilly Open Source Conference (OSCON): A major open source conference held annually in Portland. Thousands of developers and others will next converge at the Portland Convention Center July 23-27, 2007. <http://conferences.oreillynet.com/os2006/>.

Government Open Source Convention (GOSCON): The first and second annual GOSCON happened in Portland. <http://www.goscon.com/>.

Oregon Open Technology Cluster: One of the state’s targeted industry clusters; a subset of the software cluster. <http://www.oregonclusters.org/software.html#open>.

Portland Open Source Software Entrepreneurs (POSSE). <http://www.possepdx.org/>

Eclipse Foundation: Center of numerous open source projects. Eclipse is headquartered in Canada, but a key portion of its staff is here. <http://www.eclipse.org>.

Intel, IBM, Google, etc.: All are directing substantial dollars and personnel toward open source work in Oregon.

These initiatives and others represent significant open source investments in Oregon. Oregon’s entrepreneurs have long bemoaned the relatively limited number of local investors, which continues to be a problem. We will really have “made it” when no one feels compelled to recite the above litany of open source “stuff,” and we can instead move directly to advanced issues that affect the companies we work with day to day. However, it does seem realistic to hope that the open source “sector” can continue to attract a growing amount of local investment.

Oregon exceptionalists have observed that the collaborative, iconoclastic spirit of the open source movement is a perfect match for the values and characteristics of Portland’s growing creative class. If you look sideways, you can already find lots of relevant activity – from FreeGeek (hardware collaborative) and free WiFi to News4Neighbors (local news collaborative). These enterprises attract people to our region and those people in turn attract companies to our region.

“*The red-hot question is whether open source business models can be widely profitable for software enterprises beyond the operating system.*”

Against the backdrop of all this activity, Oregon lawyers have an opportunity to help lead the way in the development of the law around open technology. Already, some leading legal experts in the field are here. For example, Oregon lawyer Diane Peters is the General Counsel of OSDL, a director of the Software Freedom Law Center, and on the advisory board of Open Bar, Inc. (Open Bar is a voluntary bar association that provides information and education to lawyers about free and open source software. Local volunteers are needed.)

Okay, that’s enough boosterism. We have legal practices to worry about. That’s why this issue of the Section newsletter is filled with information to help you integrate open source issues into your practice. Open source risks and challenges most commonly arise in connection with licenses, mergers and acquisitions, and intellectual property strategizing. (See articles by Heather Meeker and Gwyn Murray.)

What is Open Source, Again?

Open source software is not shareware or public domain. It is “owned,” in the traditional sense, and is lawfully used only under license. The difference is in the license terms. Most frequently, open source software is downloaded by developers for their use, along with a license that requires no signature or acknowledgment. Many open source licenses carry requirements and restrictions that affect how the licensee may distribute or modify the software. For example, the license will terminate if its terms are violated.

The commonly accepted “Open Source Definition” is promulgated by the Open Source Initiative (<http://www.opensource.org>). OSI acts as a gatekeeper, approving and certifying open source licenses. An OSI-approved license has these key attributes, among others:

Does not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. No royalty or other fee may be required for such sale.

Allows modifications and derived works, and allows them to be distributed under the same terms as the license of the original software.

The rights attached to the program apply to all to whom the program is redistributed without the need for execution of an additional license.

The licensed program includes the source code, or tells how to obtain the source code for no more than nominal cost.

There are many different open source licenses, and each must be analyzed on its own terms. In fact, many believe there are too many open source licenses, and an effort is under way to reduce the number of them. No lawyer should try to write a new open source license unless he or she has thoroughly studied the existing field and the challenges posed by license proliferation, if then.

The most widely distributed open source software license is the General Public License (GPL), which covers Linux and many other products. The GPL (version 2) provides as follows:

“You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.”⁸

There is much debate and discussion over the proper interpretation of the GPL, particularly this provision, but there is no U.S. case law. Given the many ways in which developers can combine and assemble different pieces of software, questions often arise as to whether or not a combination that contains a piece of “GPLd” software constitutes a “work” covered by the GPL. (Yes, “GPL” has been “verbified.”)

The determination of whether a piece of software is a “work . . . that in whole or in part contains or is derived from” GPLd software can have significant consequences. If a software combination is covered by the GPL because it constitutes such a “work,” then any distribution of that combination must be licensed under the terms of the GPL. In layman’s terms, it’s no longer “proprietary” though it may be protected by copyright. Since developers love to use open source software, this issue will continue to arise more and more.

Indeed, the GPL is a pervasive piece of private law even though it was written by software developers and not by lawyers. In fact, after more than fifteen years of increasing use, the flaws and challenges of GPL version 2 have become fairly clear. A major semi-public process is under way to develop GPL version 3.

Conclusion

Open source is not just a world-challenging movement. It’s a set of software licenses that can affect how software is developed, marketed, licensed and sold. It can also affect legal analysis, drafting and advice. While the future is always uncertain, most any technology law practice is certain to face open source issues.

Endnotes

- 1 *Open, But Not As Usual*, *The Economist* 65 March 73 (Mar. 18, 2006).
- 2 Eben Moglen, *Anarchism Triumphant: Free Software and the Death of Copyright*, *First Monday* (1999), http://firstmonday.org/issues/issue4_8/moglen/index.html#m5 (accessed Nov. 13, 2006).
- 3 U.S. Const. art. I, §8, cl. 8.
- 4 Wikipedia, *Steward Brand*, http://en.wikipedia.org/wiki/Steward_Brand (last updated Oct. 18, 2006).
- 5 IDC, *The Linux Marketplace – Moving from Niche to Mainstream*, http://www.osdl.org/docs/linux_market_overview.pdf (Dec. 14, 2004).
- 6 The discussion of business models is indebted to: John Koenig, *Seven Open Source Business Strategies for Competitive Advantage*, *IT Manager’s Journal*, <http://www.itmanagersjournal.com/articles/314?tid=85> (last updated May 13, 2004); Christopher Koch, *Your Guide to Open Source Business Models*, http://www.cio.com/archive/021506/opensource_sidebar2.html (last updated Feb. 15, 2006); Richard Gorman, *Open Source Venture Capitalist Answers Your Questions*, *Slashdot*, <http://interviews.slashdot.org/interviews/06/10/13/1451242.shtml> (last updated Oct. 13, 2006).
- 7 Gorman, *Open Source Venture Capitalist Answers Your Questions*, <http://interviews.slashdot.org/interviews/06/10/13/1451242.shtml>.
- 8 OpenSource.org, *The GNU General Public License (GPL) §2(b)*, <http://opensource.org/licenses/gpl-license.php> (accessed Nov. 13, 2006).

Bringing FOSS Into The Enterprise: Thoughts On “Best Practices”



By Gwyn Firth Murray±

Matau Legal Group

www.mataulegal.com

Only a few years ago there was limited information available about how to manage Free and Open Source Software, or “FOSS”, in the enterprise – either from a business or a legal perspective. Lots of “Fear, Uncertainty and Doubt”, or “FUD”, was gravitating around about how to manage the inflow of FOSS into proprietary technology companies. Initially, my response to questions about “best practices” for managing the inflow of FOSS into one’s company – and the response of many of my colleagues also familiar with FOSS issues – was “just do what you already are doing; that is, leverage your existing intellectual property policies.” The problem with that response was that it quickly became clear that many companies had no policies or procedures in place for managing and/or monitoring the inflow, and corresponding outflow, of software IP rights. Or, if they did, they weren’t following them, regardless of whether those policies and/or procedures applied to “FOSS” or “closed source” software.¹

Software is Software

So my first point here is that “Software is Software”. In other words, companies should have policies and procedures in place for managing the intake of software and intellectual property rights. And then they should follow up and maintain them. In terms of such policies and procedures, there is a lot that is not new when it comes to FOSS. When considering what third party code they will bring in, commercial entities should be thinking about how that code fits within their overall business plan and strategy, future product line, distribution and revenue model. With those considerations in mind, companies should then carefully select those vendor products required or useful to implement that particular product development and marketing strategy. Companies should be doing a technical review and evaluation of potential incoming third party code for suitability within their own product; and reviewing (and/or negotiating)

the applicable license(s) to be sure that they have the rights they need throughout the evaluation, development, production and distribution processes. And, of course, every company should have systems in place to monitor and enforce compliance with those license terms, and to track software acquisition and development, throughout all phases of technology development, production and release.

What might an effective intellectual property management program for managing the influx, use and outflow of software look like? Elements of such an ideal program would include:

- The company’s executive team and all relevant company functions (i.e., legal, engineering, human resources, product management) not only understand the implications of software functionality and associated license rights in the context of the company’s business, but also are clear about the company’s approach to managing intellectual property rights.
- Communication between the legal and engineering teams relating to the use of incoming software is open and frequent.
- The company has a defined policy on incoming software that covers both proprietary and open source code in light of the company’s culture and business. This policy provides clear guidelines to employees (and contractors) about when and for what purposes different vendor products, including competing technologies and FOSS, can and cannot be used and when legal review is required or not.
- When in-licensing software and throughout the development and production process, the company consistently asks and considers questions such as: are we bringing this software in for evaluation purposes only? For internal development and testing? Do we intend to modify this particular software? How do we expect to link it (i.e., statically or dynamically) or otherwise combine it with other company and third party software? Are we acquiring it for internal operations purposes only? For inclusion in a product that will be licensed to end users (or via resellers or Original Equipment Manufacturers (OEM’s)) on a

“... working effectively with FOSS requires cultural change – among companies, their employees and their lawyers.”

royalty basis? Will it be used to run a back-end server supporting an ASP (Active Server Page) or other kind of services business? Do we need “downstream” rights to sublicense or further distribute? The answers to these questions help determine what inbound license terms are and are not accepted by the company.

- The company has in place a process whereby all vendor/third party code receives an appropriate level of technical and legal review for compliance with license terms and license compatibility throughout the software development and production process, including evaluation/testing, development, production and distribution (including “downstream” licensing or redistribution).
- The company has clear policies covering the use of third party code by outsourced developers and other vendors, including representations and warranties required from such vendors. These have been communicated to all company vendors as well as to the company’s purchasing and contract management organizations in advance of contract negotiation.
- The company’s IT resources are engaged so as to create a central repository for third party and “home-grown” code, and the company has purchased, or designed, a software tool to conduct and maintain a dynamic inventory of all software used by the company. This software tool tracks where and how the software is used, when and how it is modified, applicable inbound license terms, and assists the legal team in assessing license compliance and compatibility.
- The company has designated and publicized internally those individuals responsible for handling questions and approvals regarding the inflow, use and outflow of software code.
- The company has an intellectual property/software review board that consists of legal, engineering, human resources and product management staff that meets regularly and that is responsible for implementation, periodic review and enforcement of the company’s IP policy.
- Company policy and accompanying practices are periodically reviewed and consistently enforced over time.

Working Well With FOSS Requires Cultural Change

All of the above recommendations apply equally whether one is operating within the world of “closed source” software or in the world of FOSS. However, the second main point I have to make here is that working effectively with FOSS requires cultural change – among companies, their employees and their lawyers. There are vast cultural and practical differences between working with and within the FOSS “Community” and that of “closed source” software.² The chances of implementing an effective policy are increased when we as lawyers are willing to learn about FOSS, get “down and dirty” by understanding technical specifications and software functionality, and adapt to the particular cultures of our client companies and the FOSS Community.

Licensing Isn’t Just for Lawyers Anymore

Whereas lawyers used to be company “gatekeepers” without whose approval no software could come in the door, now engineers can – and do – surf the web and freely download to their computers at work and at home a wide variety of software, which comes in as both source and object code, and which is governed by a huge potential variety of license terms. When software comes in as source code, developers’ ability to modify and create derivative works from it increases, which means it is easier to commingle with proprietary code – including a company’s “home-grown” code and third party code. The consequences of such commingling – particularly with respect to FOSS code governed by “copyleft” license provisions such as those included in the GNU General Public License (the “GPL”³) – can be severe, such as triggering requirements to make company and other proprietary source code publicly available upon distribution of commingled code.

To be successful as a lawyer in this new world of FOSS, it is essential that lawyers for commercial entities recognize that there is no more ivory tower. Many software developers know – and even more think they know – much more about FOSS software and licensing than

continued on page 8

the average high tech company lawyer. Until recently, such software engineers have enjoyed a lot of freedom to download, use and play with FOSS without questions or interference from lawyers. They also have been writing licenses, sometimes without any input from the legal profession, and sometimes those licenses include technical restrictions that the average lawyer won't understand at first glance. Many developers have been working with FOSS a lot longer than we lawyers have.

Understandably, therefore, developers may resist lawyers' efforts to intervene and create policies that restrict their freedom to play with the exciting variety of software out there, or to release software they develop on the terms they choose. This places a burden on lawyers to build relationships with the engineering community (and particularly the FOSS Community) as never before. Given the vast array of software available and the sometimes bewildering license terms under which it is made available, lawyers wanting to learn about FOSS will need to ask for help from developers if we are to learn about available FOSS technology and interpret the associated variety of sometimes quite technical licensing terms. Being an effective lawyer in the world of FOSS requires a high degree of technical knowledge; start asking questions and asking for help, and you are likely to learn a lot. Again, if you demonstrate that you genuinely are interested in technology being brought in and developed, and that you are open to the company's use of FOSS and engagement of the FOSS Community, you likely will find that engineers are more receptive to your involvement.

In other words, we need to relinquish our role as "gatekeepers" that control the influx of intellectual property into our organizations, and recognize that FOSS requires that we engage in more open and collaborative lawyering. Rather than working alone in our offices or cubicles, we must walk down the hall (or fly around the globe) to work side-by-side with colleagues across functions and across borders. Face-to-face contact can be very important here. If you take time to walk around the engineering departments and/or set up brown-bag lunches or other meetings with engineers to discuss FOSS issues, you start dispelling the notion that lawyers are the distant enemy.

Another reason for making friends with software engineers is that the amount of software available, and corresponding number of licenses out there to consider, is truly daunting. When I checked recently, the number of FOSS projects hosted on sourceforge.net was more than 130,000, with the number of registered users listed as well over 1.4 million.⁴ And Sourceforge is not the only place where FOSS is found. FOSS and other freely available software can be downloaded from other open source software repositories, commercial proprietary software companies (such as Microsoft, Sun and Oracle)⁵, open source projects (such as JBoss⁶ and PostgreSQL⁷), university and personal websites, and commercial open source companies (such as MySQL⁸), among other sources. While many of these software projects release their software under relatively well-known and well-understood licenses such as the GPL or Berkeley Software Distribution license (BSD license), many of them release software under different license terms that are referred to here as "FOSS" but may not meet accepted definitions of "open source".⁹ Some software is downloadable without a clear path to the applicable license terms, or possibly with no license terms at all. The Open Source Initiative currently lists close to sixty "OSI certified" licenses,¹⁰ but many software developers have chosen to issue their software under terms they have crafted themselves or created by modifying existing licenses – and that do not appear on the OSI list. And many of these developer-written licenses get very technical. For example, one must understand how the particular applications work together, how the programs "link" to libraries, how the software is written and compiled – in order to understand what you are permitted to do under many FOSS licenses.

But don't be scared. The same multitude of different licenses exists in "closed source" land as well – it's just that the terms aren't out there for everyone to see, and those terms generally have been written by other lawyers and sound more familiar to us as "legalese". Further, though "closed source" licenses have companies and their lawyers behind them, they generally don't have a passionate community around them to help back up enforcement efforts. In contrast, the FOSS Community

... we need to relinquish our role as "gatekeepers" that control the influx of intellectual property into our organizations, and recognize that FOSS requires that we engage in more open and collaborative lawyering.

sees as part of its mission the interpretation and enforcement of FOSS licenses, and this interpretation and enforcement takes place via often very blunt dialogue on public websites. So, here again, building and maintaining relationships with FOSS developers and the FOSS Community can be helpful. If you and your company are known to the FOSS Community as responsible licensees that endeavor to comply with FOSS license terms, the “flames” posted on FOSS-related websites may be kinder and gentler.

You Need to Read – and Re-Read – Each and Every License

So how do you deal with this daunting variety of highly-technical licenses? For a start, read the ones that apply to your software product, and read them carefully. If you don’t understand what they mean, or what they permit, find your closest friendly software developer or IT professional and ask for help understanding how the software works, how it links, how it compiles, and how it is distributed from a technical perspective. Ask “stupid questions” and you, likely, will find out that your questions aren’t so stupid. Ask other lawyers how they have handled the same kind of situation before and, likely, you will find that they are struggling with the same or similar issues. Post a question on the general discussion list at www.open-bar.org and see what comes back. If it’s a “flame”, push back on the sender and then contact the Open Bar site administrator or me. One of our key goals at Open Bar is to promote open dialogue among lawyers and legal professionals.

In my experience, most companies think the GPL is their worst nightmare because it is both widely-used and has strong “copyleft” attributes. And they are afraid of the GPL with some reason, particularly as the license is now under revision and new and different terms may apply to software licensed under that future version.¹¹ But, again in my experience, the GPL (at least in its current form) is not the FOSS lawyer’s greatest problem. Most software developers by now understand the “copyleft” implications of the GPL and how to effectively navigate around those.

I find that more companies get into trouble with respect to other licenses, whether FOSS or “closed source.” And the main reason they do is because they have not had a process in place for keeping proper records of which license applies (i.e., what license was in effect on the date the software was downloaded?), where that license is (is a copy still available on the web? Does the company have a hardcopy stored?), and making sure

that the permissions under the license continue to match the company’s needs through out the development, production and distribution process. More and more engineers understand the implications of downloading FOSS software and have read – and care enough to abide by – the terms of both their employment agreements and the applicable license before they do so. However, even the best-informed and best-intentioned developer may forget about license terms that permitted testing and evaluation but prohibit modification or distribution by the time they start incorporating the third party code and using it to develop product. Again, “Software is Software”, and here is where a well-thought-out IP management program helps keep you out of trouble.

“Ask “stupid questions” and you, likely, will find out that your questions aren’t so stupid.”

Other areas where I see companies get into trouble include forgetting about outsourcing and not considering what “distribution” means under the terms of a given license. Companies may be diligent about their own intake of third party software but then fail to monitor such intake and use by the outsourced consultants they have hired to develop their software. Companies should have in place – and purchasing and contract management staff should be familiar with, and equipped to enforce – policies and requirements around vendors’ use (or non-use) of FOSS in any products or services coming into the company. Similarly, companies should be alert to the question of what might be considered “distribution” within their own entity or to agents acting on their behalf. For example, some licenses may explicitly provide that offering software on an ASP basis equals distribution, thus triggering the reciprocity requirements of that license. Even if the license is silent on whether running an ASP service equals a “distribution”, if a multi-national corporation has affiliated companies and uses “back-end” software developed at one affiliate to provide consulting services or run an ASP offering at a sister company, is that a distribution? Does this analysis change if the affiliated company is a joint venture that is only partially owned by the parent? What about the company’s use of outsourced consultants to develop its software? If software is sent back and forth between the

continued on page 10

outsourcer and the client company during the development process, is that distribution?

Asking these questions may not lead to clear answers in any particular case. However, having good relationships and open communication with FOSS developers and other lawyers will only help you when you need to “brainstorm” and come up with creative solutions.

Conclusion

There are valid reasons that “Fear, Uncertainty and Doubt” linger around the use of FOSS in the commercial enterprise. Working with FOSS requires that we acquire more technical knowledge and be more diligent about implementing and maintaining “best practices” programs than we may have in the past. And it also means that we must build relationships with our clients and other lawyers as never before. If we are to be effective, we need to be open to help from developers who may have more in-depth knowledge of FOSS technology and licensing terms than we do. To figure out answers to legal questions when almost no statutory or case law on FOSS exists, we need to be open to brainstorming with other lawyers – even if they are our competitors. But what goes around ultimately does come around, in my view. When we are open to asking for help and to learning from the FOSS Community and from each other, the FOSS Community – and our fellow lawyers -- likely will be more open to working with, and learning from, us. In sum, Open Source requires Open Lawyering.

Footnotes

- ± Gwyn Firth Murray was one of the “early adopters” in the legal community of a focus on open source software. She is founder and principal of the Matau Legal Group, which offers a broad range of commercial, licensing, and other legal services to both start-up and established companies in the high tech and biotech industries (see www.mataulegal.com). Ms. Murray is co-founder of Open Bar, Inc., a not-for-profit organization focused on legal rights and responsibilities in the world of open source software (see www.open-bar.org) and currently serves as Vice-Chair of the Open Source Committee of the American Bar Association Section of Science and Technology. Ms. Murray is a graduate of Stanford University Law School and also holds an M.A. in Latin American Studies from Stanford University. She obtained her B.A. magna cum laude and with distinction in economics from Yale College.
- See generally Bradford L. Smith and Susan O. Mann, Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?, 71 U. Chi. L. Rev. 241 (Winter, 2004).

- 2 For more information on this topic, please see the following papers written by the author: “Free and Open Source Software: An Introduction” (January 2006) and “Getting with the Program: A Guide for Lawyers Working with Open Source Software in the Enterprise (October, 2005), both available at www.mataulegal.com and www.open-bar.org. This article is derived in large part from the latter paper.
- 3 See <http://www.gnu.org> (accessed on October 25, 2006).
- 4 See <http://sourceforge.net> (accessed on October 12, 2006, when it listed 131,830 Registered Projects and 1,412,096 registered users).
- 5 See microsoft.com, sun.com and oracle.com generally.
- 6 See <http://labs.jboss.com/portal> (accessed on October 25, 2006).
- 7 See <http://www.postgresql.org> (accessed on October 25, 2006).
- 8 See <http://www.mysql.com> (accessed on October 25, 2006).
- 9 Various definitions of what “open source” means exist. One of the most commonly referenced is that of the Open Source Initiative, which can be found at <http://www.opensource.org/docs/definition.php>. Readers may also be interested in the discussion of the difference between “open source” and “free software” found at <http://www.gnu.org/philosophy/free-software-for-freedom.html>.
- 10 <http://www.opensource.org/licenses> (accessed on October 12, 2006).
- 11 I believe that this topic is covered in another article in this newsletter, so I will not go into detail about it here.

Open Source in M&A and Other Transactions



By Heather Meeker
Greenberg Traurig LLP
E. Palo Alto, CA
meekerh@gtlaw.com

This article is an excerpt from a book in progress.

The importance of open source in mergers, acquisitions and other transactions has grown with the popularity of open source, and with the adoption of open source by commercial technology developers. In the mid-1990's, when open source was relatively unknown and considered within the purview of hobbyists, most agreements for technology transfer, acquisition, licensing or development sought primarily to exclude open source from the relevant developments or products. However, as open source has become more popular, particularly in the realm of commercial technology development, the treatment of open source in commercial agreements, mergers and acquisitions has necessarily become more sophisticated, and more complex.

This change reflected a precipitous change in the attitude of business lawyers advising technology clients. In the mid-1990's, open source was generally considered too dangerous to include in commercial products. The attitude of intellectual property lawyers during that time could roughly be summed up in the parental admonition: "Don't put that in your mouth -- you don't know where it's been." As it became clear that open source was coming into common use in commercial developments, this attitude necessarily changed, and in turn there has been a change in the treatment of open source in transactional agreements.

In the mid-1990's, representations like the following began to appear in merger, acquisition, or investment agreements: (At that time, contracts relating to routine technology development and licensing generally contained nothing about open source.)

The Company Intellectual Property contains no software source code that is covered by a so-called "open source," "copyleft" or "general public" license. For purposes of this paragraph, "open source" is any software that is made generally publicly available in source code form.

This representation demonstrates a few of the problems that arose in negotiating agreements during a time of minimal awareness about open source. The definition of "open source" above is overbroad, demonstrating a lack of understanding regarding the nature of software source code. Many kinds of software are, of course, available in source code form regardless of the type of license applied to them. Not all software is compiled. In fact, large categories of software, such as scripting languages and markup languages, are executed without an intermediate compilation and are thus never available in any form other than source code form. The definition above captures all uncompiled languages such as PERL, HTML, or interpreted BASIC. Open source, instead, is more sensibly applied to software that could be distributed only in compiled or binary form, but is voluntarily made available in source code form.

As attorneys became more familiar with open source, the representations they drafted began to look more like this:

The Company Intellectual Property contains no software source code that is covered by a so-called "open source" license. For purposes of this paragraph, "open source" is any software that is made generally publicly available under licenses approved by the Open Source Initiative.

Lawyers by nature love objective definitions, and at least on its face, the open source definition seemed a useful one for such a purpose. However, this approach is too narrow. The Open Source Initiative does not approve all licenses that fit the definition, only licenses that are submitted to them for approval. This was true even before the organization narrowed its criteria for approval of licenses in 2006,¹ because there are many variations of licenses such as BSD, MIT, and Apache that are not on the official list of approved licenses, but are generally thought to be open source, and fit the open source definition.

Of course, representations like this, soon, implicitly acknowledged that an unqualified statement excluding open source was rarely true. At that point, representations were redrafted to compel disclosure of open source, rather than simply excluding it. This is accomplished by a trivial variation as follows:

continued on page 12

“However, over time – as the 1990’s gave way to the 21st century – it became clear that the primary risks of open source surrounded compliance with open source licenses, rather than concerns about third party intellectual property infringement.”

Except as set forth in the Disclosure Schedule, the Company Intellectual Property contains no software source code that is covered by a so-called “open source” license.

Such a representation was designed to elicit disclosure of all the open source software a company was using in its business. Although this was a step in the right direction, it was not very nuanced. Such a broad representation at the same time requested more information than the acquirer cared to know, and not enough.

Such a representation sought disclosure of all open source used in the business, rather than all open source contained in the company’s products. The failure to make this distinction generally sprang from an assumption, which is now more widely thought to be misguided, that the primary liability and risk associated with open source was the possibility that, due to the number of its contributors, open source was likely to infringe third party copyrights. If this were the case, it would be nearly as important to know all open source used in a business as it would be to know all open source contained in a company’s products. However, over time – as the 1990’s gave way to the 21st century – it became clear that the primary risks of open source surrounded compliance with open source licenses, rather than concerns about third party intellectual property infringement.

Thus, the representation asked for information that might not be very useful – namely a list of open source that was used in the business but not incorporated into products. This would compel a disclosure of the use of common open source tools like the Apache web server, as well as every open source product ever downloaded by any employee and used on his desktop. It is probably fair to say that no merger, acquisition or investment was ever sidelined or revalued based on the use of open source code on an end-user basis. At the same time, a representation like the one above does not compel disclosure of important information necessary to analyze the compliance of the company with its inbound open source licenses for software integrated into company products.

This gave way, in the early 2000’s, to a more sophisticated approach, in two ways. First, it was understood that open source simply used in a business was of dramatically less concern than open source embedded in products. Second, it was understood that open source that was provided under permissive licenses was of far less concern than open source provided under hereditary licenses like GPL, LGPL and MPL. Although custom and practice in this area is still developing, the following is an example of an open source representation that might typically be found in a merger, acquisition, or investment agreement today that takes a more sophisticated approach to the topic.

Disclosure of Facts Regarding Open Source Software

Section _____ of the Disclosure Schedule: (a) lists all Open Source Software that at any time prior to the Effective Date has been used in the conduct of the Business; (b) describes how each element of such Open Source Software has been used in the conduct of the Business; (c) states whether such Open Source Software has been distributed to any third party, and in the case such Open Source Software has been so distributed, describes such distribution; and (d) states whether such Open Source Software has been modified by or for Company. For purposes of this paragraph, “Open Source Software” means all software licensed, to the Company or by the Company to third parties, under licenses substantially similar to those approved by the Open Source Initiative and listed at <http://www.opensource.org/licenses/>, which licenses include without limitation the GNU General Public License, the GNU Lesser Public License, the Berkeley Science Division (BSD) License, and the Apache License.

Of course, this representation also focuses on open source that is used rather than distributed. If one wishes to dispense with this risk and focus on compliance issues, the following representation would be useful.

Disclosure of Facts Regarding Viral and Non-Free Software

Section ____ of the Disclosure Schedule: (a) lists all Open Source Software that at any time prior to the Effective Date has been included in any Company Product; (b) states whether such Open Source Software has been modified by or for the Company; and (c) for all Proprietary Software that is distributed in connection with or interoperates with any Free Software, describes the interfaces between such Proprietary Software and Free Software (such as dynamic or static linking, forking, or communications protocol interfaces), and lists the names of any related published APIs. As used in this paragraph, the following terms have the following meanings:

“Open Source Software” means any software that is generally available to the public in source code form under licenses substantially similar to those approved by the Open Source Initiative and listed at <http://www.opensource.org/licenses/>, which licenses include without limitation the GNU General Public License, the GNU Lesser Public License, the Berkeley Science Division (BSD) License, and the Apache License.

“Free Software” means any Open Source Software licensed under a license agreement that requires, as condition of being distributed or otherwise, that the source code for the Open Source Software or any derivative works thereof be made available to licensees to whom such software is distributed, including without limitation any software licensed under the GNU General Public License, the GNU Lesser General Public License, the Mozilla Public License, the Common Public License, or the Common Development and Distribution License.

“Proprietary Software” means any portion of the Company Product that has been distributed by Company in object code form only (or that is available in source code form only upon the occurrence of an event such as cessation of business or business distress, such as part of a software source code escrow).

Contrast the following representation:

Open Source Representation – Compliance

To the extent Open Source Software has been incorporated into any Company Product or otherwise used in the conduct of the Business, Company and its distributors, resellers, and licensees are in compliance with the terms of any Open Source Software licenses governing the use of such Open Source Software.

This representation, straightforward on its face, may effectively be a legal opinion rather than a disclosure of facts, and is thus risky for the seller and its counsel.² This representation may also be subsumed by more general representations of compliance with inbound intellectual property licenses, of the type found in most merger and acquisition agreements.

Open source in licensing and commercial transactions

The above discussion focuses on management of risk regarding the use of open source code. Provisions in transactional agreements that seek to manage liability look largely the same in mergers, acquisitions and investments as they do in commercial transactions. Thus, today, many software licenses and development agreements contain open source representations like the ones above.

However, license and development agreements also require provisions particular to open source that go beyond representations and warranties. The primary example is the language necessary to integrate open source code that is provided under hereditary licenses with that provided under nonhereditary licenses or commercial terms. For instance, if a company, Licensor, is licensing a product that contains both open source and proprietary components, the Licensor must insure that its commercial binary license agreement does not unintentionally violate the inbound hereditary licenses that apply to open source components of the product. (Code covered by permissive licenses can, of course, be re-licensed under binary terms.) For instance, the GPL contains the following provision:

The GPL says:

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the

continued on page 14

original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein....³

To avoid this problem, any commercial end-user license covering a product that contains some components covered by hereditary agreements should contain a provision such as the following:

“The attitude of intellectual property lawyers during that time could roughly be summed up in the parental admonition: “Don’t put that in your mouth – you don’t know where it’s been.””

Notwithstanding the foregoing Section [reference license grant], Licensee acknowledges that certain components of the Software may be covered by so-called “Open Source” software licenses (“Open Source Components”), which means any software licenses approved as open source licenses by the Open Source Initiative or any substantially similar licenses, including without limitation any license that, as a condition of distribution of the software licensed under such license, requires that the distributor make the software available in source code format. Licensor shall provide a list of Open Source Components for a particular version of the Software upon Licensee’s request. To the extent required by the licenses covering Open Source Components, the terms of such licenses will apply in lieu of the terms of this Agreement, and Licensor hereby represents and warrants that the licenses granted to such Open Source Components will be no less broad than the license granted in Section [____]. To the extent the terms of the licenses applicable to Open Source Components prohibit any of the restrictions in this Agreement with respect to such Open Source Component, such restrictions will not apply to such Open Source Component.

Now that the licenses are sorted out, the licensor must choose whether other provisions of its end-user license agreement will cover the open source components. Most end-user license agreements contain provisions such as performance warranties (warranties that

the software will perform according to specifications), maintenance and support obligations, intellectual property infringement indemnities, and the like. Hereditary open source license agreements permit a re-distributor to make additional warranties and covenants about the software they cover.⁴ It is a business strategy decision, however, whether to include open source software under these commercial terms or not.

Most commercial end-users will demand that all software included in a product, whether open source or not, be subject to performance warranties. Some licensors are successful in excluding open source code from intellectual property warranties and indemnities, but many are not. While licensors can make a convincing case that they cannot control whether open source code violates third party rights, licensees who are paying commercial fees for a license to a product often take the position that it is the licensor’s decision to include open source in the product and therefore the licensor’s burden to bear liability for possible infringements, the theory being that the licensor is in a better position than the licensee to take a reserve against expenses associated with intellectual property infringement, and that the license fees for the product should be used to fund that reserve.

Development Agreements

The popularity of open source has bred a new kind of development agreement -- one in which the material developed is primarily covered by open source licenses, instead of the work-for-hire arrangement that is customary in commercial software development. When developers are hired to modify software that is covered by hereditary license agreements, the deliverables that they provide consist primarily of existing open source code that cannot be provided under any agreement but the hereditary one. The rights in newly developed code could be assigned by the consultant to the customer, but: (1) the customer would regardless be bound to the terms of the license for most of the code delivered; and (2) developers increasingly take the position that modifications they make to open source code should be shared

with the community. In this way developers, perhaps fortuitously, are solving a perennial problem that developers face when doing sequential development projects: the ability to reuse past work in future engagements. While many consulting agreements in the proprietary context contain provisions that allow for the delivery of third-party code, these provisions have generally been considered exceptions rather than the rule. Thus, the intellectual property ownership provisions in an open source development contract might look like this:

Deliverables. All copyrightable material, software, documentation, and other works of authorship embodied in materials delivered to Customer hereunder (“Deliverables”) will be provided to Customer under the terms of the open source license agreements described in Exhibit ____.

Third Party Materials. For all materials designated as “Third Party Materials” on Exhibit B, the parties acknowledge that such materials will be necessary for Customer to use the Deliverables, and Customer will be solely responsible for obtaining necessary licenses to the Third Party Materials. Consultant may provide copies of the Third Party Materials as a courtesy, however, the licenses to such materials will be as specified in Exhibit ____.

We Are All Teachers

Open source issues in transactions can be complex, but the key to working through them is understanding and knowledge. Different companies take different approaches to bearing risk of open source intellectual property infringement, and license compliance. However, those differences represent a much narrower divide than the one between parties knowledgeable about open source and those that are still struggling to understand it. Today, anyone doing deals in the technology space needs a basic understanding of this growing area. If you find you are the party in the transaction who has the knowledge, that is your cue to share it.

Endnotes

1. S.J. Vaughan-Nichols, *Open-Source Referees Change the Rules*, <http://www.eweek.com/article2/0,1895,1783791,00.asp> (April 7, 2005).
2. A discussion of the legal uncertainties of GPL compliance is beyond the scope of this topic, but see *supra*.
3. Open Source Initiative, *The GNU General Public License (GPL)*, <http://www.opensource.org/licenses/gpl-license.php> (2006).
4. The GPL states, “You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.” *Id.* at <http://www.opensource.org/licenses/gpl-license.php..>

Implied Patent Rights and Open Source Software



By Michele K. Herman
Woodcock Washburn LLP
mherman@woodcock.com



By Peter M. Ullman
Woodcock Washburn LLP
pullman@woodcock.com

There has been extensive discussion and analysis of the various “copyleft” provisions found in some Open Source Software (OSS) licenses. Copyleft provisions are the terms in some OSS licenses that grant a party the right to use, modify, and redistribute the software – but only if the party grants all downstream users the same rights in any modified version of the software that the party distributes. Copyleft provisions require a party who modifies the software to distribute source

code for some or all of the modifications that the party makes to the software. This latter feature of copyleft often prevents modifications to open source software from being maintained as a trade secret, and the effect of OSS licenses on trade secret protection is well-understood by the software community. What is not as well understood is the effect that OSS licenses may have on a company’s patent rights.

continued on page 16

Most companies that use open source software carefully evaluate the software licensee's obligations in order to determine how the licenses will affect the company's ability to protect the software that company develops as a trade secret. Yet far fewer entities evaluate these licenses to ascertain whether they unknowingly grant the patent rights with the distribution of such software. These entities often have large patent portfolios that are just as important to their business models as are trade secrets, so it is important for such entities to understand what patent rights they may grant when they distribute OSS.

Implied Patent Licenses to Use, Make or Sell Software

Software is normally protected by at least a copyright. Thus, most software licenses are essentially copyright licenses for use of the software. OSS licenses are, in this regard, no different from ordinary software licenses. Since open source software is normally copyrighted, the OSS license simply grants the licensee the use of, and certain other rights in, the copyrighted software. The copyleft provisions that are featured in many OSS licenses are simply a way of using copyright law to ensure that certain provisions of the OSS license, e.g., the right to distribute for free and the requirement to release source code, will carry through to downstream parties who distribute or modify the licensed software. (For a general discussion of how copyleft works, see the Wikipedia article on copyleft.¹) Thus, an OSS license typically contains a grant of the right to "use" the copyrighted software.

What is often overlooked is that the grant of a right to "use" software may give rise to an implied license for certain patents that the distributor of the OSS owns. It has long been recognized that "[a]ny language used by the owner of the patent, or any conduct on his part exhibited to another from which that other may properly infer that the owner consents to his use of the patent in making or using it, or selling it, upon which the other acts, constitutes a license"² Granting the right to "use" the software could be understood as an intent to grant the licensee any patent rights that the patentee holds and that would be needed to use the software.³ Thus, a right to "use" the software may create an implied "use" license under any patent claims that the licensor owns or has the ability to license.

Moreover, if the license contains a copyleft provision, then the licensor essentially grants all downstream users the right to use the distributed software, or *modified versions of the software*. So, when OSS is subject to

a copyleft provision, any distributor of that software may be granting an implied patent license for any modified version of the distributed software that is created later on. Such a broad grant may be prevented if the license contains an express grant or reservation of patent rights. We discuss this point later. In theory, a downstream contributor could modify a software work by adding a host of features that are covered by a prior contributor's patents, and then argue that these new feature are now implicitly licensed under those patents.

One issue to consider is that a patent license – even an implied patent license – may grant certain rights but not others, so it is important to read the language in a given license to determine which patent rights, if any, might be implicitly granted. A patent grants the patentee the right to exclude others from using, making, or selling the patented invention.⁴ Some OSS licenses may imply a patent right to "use" or "sell" the software, but not the patent right to "make" the software. For example, both the Apache 1.1 license, and the BSD license contain no express grant of patent rights, but do contain the following general permission to use the software:

Redistribution and use in source and binary forms, with or without modification, are permitted⁵

"Redistribution" might imply a right to sell any patented invention contained in the software. Moreover, to "make" a software product generally means creating a usable copy, so it is notable that the Apache 1.1 and BSD licenses do not explicitly grant a right to reproduce or copy the software. In contrast, the Python license – another license that is silent on the subject of patent rights – grants the right to "reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use" the software.⁶ Thus, the Python license could be understood to grant a patent right to "make" the software, even if the Apache and BSD license do not. It is interesting to note that these licenses may be viewed as granting different implied patent rights, although, in practice, the distinction may not be of much practical significance, since one could download multiple copies of the software and then redistribute the downloaded copies.

The Apache 1.1, BSD, and Python licenses are all examples of non-copyleft OSS licenses. In other words, they do not require redistributors of the licensed software to grant parties the same rights that the redistributor received. Thus, while the party who receives software under Apache 1.1, BSD, or Python may receive source

“What is often overlooked is that the grant of a right to “use” software may give rise to an implied license for certain patents that the distributor of the OSS owns.”

code and (implied) patent rights, that party can redistribute the code, or any modifications, under different terms. So, a party who receives software under the Apache 1.1 license, for example, might redistribute a modified version of that software under a new license that expressly grants (or withholds) any “make” or “use” of patent rights. Since there are no copyleft provisions, the party could even choose whether or not to distribute the source code for the modifications.

Thus, a party who receives software under an Apache 1.1, BSD, or Python license can avoid granting implied patent rights, or divulging of trade-secret code modifications, simply by redistributing the software under different license terms. However, if that party were to choose to redistribute the software under the same license, the party may be granting implied patent rights.

The GPL version 2.0 is carefully worded to avoid conflating the grants in the copyrighted OSS with patent rights such as “to use” as some of the licenses discussed above.⁷ However, Section 6 of the GPL version 2.0 states:

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions.⁸

As noted above, licenses that grant the copyright permissions to the rights to “copy” and “distribute” (or “redistribute”) software may imply the patent rights to “make” and “sell” patented inventions contained in the software. Therefore, if you are an original licensor under the GPL,⁹ you may have implicitly granted the patent rights to “make” and “sell” copies of the program you originally distributed, and also any subsequent modifications that others made to the program.¹⁰ The licensor potentially licenses an unlimited number of downstream recipients the rights to “make” and “sell” any software that is “based on” the software distributed under the GPL by that licensor. Without clear legal precedent to help interpret the impact of these copyright grants, there are widely differing views regarding the extent of the implied patent license risk.

Unlike the Apache 1.1 and BSD licenses, the GPL version 2.0 is a copyleft license and a recipient of OSS under the GPL cannot avoid implied licenses simply by redistributing the OSS under different license terms. A recipient of software under the GPL who redistributes that software must apply the GPL to the redistributed software. In the authors’ opinions those desiring the freedom to license their patents under commercial terms or not license them at all should carefully weigh the risks of such implied patent licenses before distributing any software under the GPL version 2.0.

Effect of Express Grant of Patent Rights in Software Licenses

Some OSS licenses include an express patent license. An express grant of patent rights may foreclose or limit any implied patent rights that would otherwise arise. In *Intel v. Broadcom*,¹¹ a federal court in Delaware observed that “where an agreement contains a specific provision expressly defining the scope of the patent license implied licenses dealing with the same subject matter are not generally recognized.”¹² The court noted the general principle that “[t]here cannot be a valid express contract and an implied contract, each embracing the same subject matter, existing at the same time.”¹³ Thus, an OSS license that contains an express grant of some patent rights may prevent other implied patent rights from arising.

Express patent grants take various forms. When an express license is present, a party that distributes software generally grants all downstream parties a license to any patent claims embodied in the software that the distributor either contributed, or passed along from a prior contributor, depending upon the license. However, express patent grants in OSS licenses generally do not extend to features that are subsequently added by other parties. For example, the Open Software License version 3.0 (OSL 3.0), gives the licensee a license under “patent claims owned or controlled by the Licensor that are embodied in the Original Work as furnished by the Licensor”¹⁴ Different wording is used in other licenses such as the Apache 2.0,¹⁵ Mozilla 1.0,¹⁶

continued on page 18

Mozilla 1.1,¹⁷ and Eclipse 1.0,¹⁸ but the substance of the patent grant in these licenses extends only to software as actually distributed by the patent holder, not to future modifications.

The latest draft of version 3.0 of the GPL¹⁹ attempts to address some of the questions raised in version 2.0 regarding implied patent rights by including an express covenant not to assert essential patent rights against downstream recipients of the software. It is unclear whether or not a covenant not to assert patents will disclaim an implied patent license, as an express grant of patent rights would seem to do. In any case, many licenses that include express patent grants also include a provision clarifying that no other licenses are granted by implication. This latest draft of the GPL instead states:

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

Should the covenant and the above quoted language remain in the final draft of the GPL Version 3.0, those intending to use OSS under this license should very carefully consider what patent rights may be impliedly granted to others with the use or distribution of the OSS whether unmodified or modified.

Defensive Termination Provisions

An increasing number of software licenses – especially OSS licenses – include defensive termination provisions. Under these provisions a party's license to redistribute or even use OSS could be terminated if the party asserts a patent claim against another party in court. The scope and triggering mechanisms for such termination provisions vary widely among licenses. Some popular OSS licenses that include such defensive termination provisions include the Apache License version 2.0, the Eclipse Public License, and the Mozilla Public License version 1.1.²⁰

Although a defensive termination provision is not likely to have the legal status of an express or implied patent license, it could be tantamount to one. The licensee's right to use the OSS may terminate if the licensee enforces its patent against certain parties. So, if the OSS is critical to the party's operations or business, then a defensive termination provision essentially requires that party to allow others to use its patented technology.

Conclusion

As the popularity of OSS grows, its inclusion in commercial products will become mainstream among companies with important patent portfolios. Companies that include OSS in their commercial offerings or that use OSS as an important part of their commercial operations or business model may need to consider how that use may affect the company's patent portfolio, particularly if the company has portfolio licensing programs. While such use of OSS is well-known to increase risks associated with trade secrets, less attention is often paid to the broad implied patent grants that could result from the use or distribution of certain OSS under some circumstances. Companies that plan to use or distribute OSS should make sure that they understand, in advance, what effect the OSS license will have on the company's patent portfolio.

Endnotes

1. <http://en.wikipedia.org/wiki/Copyleft> (last modified Oct. 26, 2006).
2. *Wang Labs. v. Mitsubishi Elecs. Am.*, 103 F.3d 1571, 1580 (Fed. Cir. 1997) (quoting *De Forest Radio Tel. Co. v. U.S.*, 273 U.S. 236, 241 (1927)).
3. *Cf. AMP, Inc. v. U.S.*, 389 F.2d 448, 453 (Ct. Cl. 1968), cert. denied, 391 U.S. 964 (1968) ("There is ample authority for the rule that where the owner of a patent grants to a licensee the right to use a patented machine, the grant carries with it, by necessary implication, a license under any other patent of the licensor which would be infringed by operation under the grant.") (quoting *Stevens v. Steel & Tube, Inc.*, 114 F.2d 815 (6th Cir. 1940)).
4. See 35 U.S.C. § 271(a) (2006).
5. Apache Software License 1.1, <http://www.opensource.org/licenses/apachepl.php> (accessed Oct. 30, 2006) and the BSD License, <http://www.opensource.org/licenses/bsd-license.php> (accessed Oct. 30, 2006).
6. Python License, section 2 (emphasis added), <http://www.opensource.org/licenses/pythonpl.php> (accessed Oct. 30, 2006).
7. "Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does." GPL version 2.0, section 0, <http://www.opensource.org/licenses/gpl-license.php> (accessed Oct. 30, 2006).
8. *Id.* at section 6.
9. The term "original licensor" is not defined by the GPL. Several different interpretations are possible. The extent of the implied patent licenses may vary substantially depending on the interpretation.
10. It would be difficult to find an implied right to "use" the modified or unmodified Program from the language

of the GPL itself. However, if the distribution of the Program constitutes a "sale," then the doctrine of patent exhaustion may apply. Patent exhaustion may enable a downstream recipient to use a copy it receives without having to obtain a patent license from the original licensor.

11. *Intel Corp. v. Broadcom Corp.*, 173 F. Supp. 2d 201 (D. Del. 2001).
12. *Id.* at 213.
13. *Id.* (citing *Wal-Noon Corp. v. Hill*, 45 Cal. App. 3d 605, 613 (Cal. App. 3rd Dist. 1975)).
14. OSL 3.0, section 2 (emphasis added), <http://opensource.org/licenses/osl-3.0.php> (accessed Oct. 30, 2006).
15. Apache License version 2.0, section 3, <http://www.opensource.org/licenses/apache2.0.php> (accessed Oct. 30, 2006).
16. Mozilla Public License version 1.0, sections 2.1(b) and 2.2(b), <http://www.opensource.org/licenses/mozilla1.0.php> (accessed Oct. 30, 2006).
17. Mozilla Public License version 1.1, sections 2.1(b),(d) and 2.2(b),(d), <http://www.opensource.org/licenses/mozilla1.1.php> (accessed Oct. 30, 2006).
18. Eclipse Public License version 1.0, section 2b, <http://www.opensource.org/licenses/eclipse-1.0.php> (accessed Oct. 30, 2006).
19. As of the time of this writing, the second draft of version 3.0 has been released. However, a third draft could be released according to Free Software Foundation in October 2006. See <http://gplv3.fsf.org/process-definition#SECTION008000000000000000> (last modified Jan. 15, 2006).
20. These licenses can be viewed at <http://www.opensource.org>.

Harald Welte, Linux, and the GPL



By H. Tomás Gómez-Arostegui

Assistant Professor of Law

503-768-6816

tomas@lclark.edu

Introduction

Those familiar with open-source software and the GNU General Public License (GPL) know that no court has expressly ruled on the enforceability of the GPL under U.S. contract law. Not long ago, in the pages of this newsletter, Duke Tufty demonstrated as much when he summarized the domestic developments in this area.¹ That summary has largely stood the test of time.² Tufty did not, however, with the exception of mentioning a 2004 decision from Germany,³ endeavor to cover international enforcement of the GPL. That task has been left to me.

Unfortunately, there is little to report, at least by way of legal decisions. It appears the GPL has only been tested abroad on three occasions. All three cases were brought in German courts and analyzed under German law. And, not surprisingly, the cases always involved the Linux operating system.

The first was the 2004 case, which Tufty has already mentioned. That case upheld the validity of the GPL in the course of granting a preliminary injunction. The Court enjoined the defendant, a manufacturer of networking

hardware, from distributing hardware embedded with portions of the Linux software without also making the source code available free of charge. The decision to grant the preliminary injunction was upheld on appeal,⁴ and it appears the defendant acquiesced.

The second instance occurred in April 2005, when the Munich District Court granted a similar preliminary injunction against Fortinet Ltd., ordering that the company not distribute its products until it became GPL compliant.⁵ Following the entry of the injunction, the parties reached an accommodation, and the defendant agreed to make its source code available to those who requested it.⁶

Because both decisions were made on a preliminary record, without the benefit of a full hearing, some German scholars have questioned the precedential value of the decisions.⁷

Just two months ago, however, in September 2006, another decision appeared, this time after a full hearing in the Frankfurt District Court. The Court rejected an argument that the GPL was not legally binding on a company that had distributed Linux code as part of a network storage device.⁸ The decision has been heralded as a major victory for the GPL.⁹ But the judgment may still be appealed, so it remains to be seen whether the outcome, or the district court's reasoning, will be upheld.

continued on page 20

These three decisions, having been made in Germany and reasoned according to German law, are not likely to generate much interest in U.S. courts, beyond perhaps serving as comparative or anecdotal citations. That having been said, U.S. companies who do business in Germany, or hope to do so, would be advised to consider the decisions, regardless of the procedural posture in which they were decided.

But there remains a more important reason to be aware of these decisions—one that has less to do with the decisions themselves and more to do with the person behind them. The plaintiff in each case has been Harald Welte, a 27-year old software programmer in Germany. Welte, along with his lawyer Till Jaeger,¹⁰ runs a GPL enforcement project in Germany called GPL Violations.¹¹

According to Welte's website, the main purpose of the project is the "gathering, maintaining and distributing [of] information about people who use and distribute GPL licensed free software without adhering to the license terms."¹² The software covered by the GPL that Welte polices has, for the moment anyway, always been Linux.

Since creating the project in 2003, Welte has filed 3 lawsuits in an effort to maintain and support the GPL—the German cases mentioned above. He claims to have obtained pre-litigation settlements in at least 40 other cases.¹³ Some of the vendors he contends he has set straight include big names such as Siemens, Belkin, and NetGear.¹⁴

In the paragraphs that follow, I aim to give a sense of how it is that Welte can sue GPL violators, what it is that he wants, and why U.S. companies and their lawyers should care.

I.

Those of you already familiar with the open-source movement will know of Richard Stallman. In 1984, Stallman began the process of creating an operating system (OS) that would be compatible with Unix, but that would be based entirely on code that could be freely and openly exchanged. His OS was called GNU, which is a recursive acronym for GNU is Not Unix. To support his project, he drafted the first open-source license, which he called the GNU General Public License, or the GPL.¹⁵

Stallman's OS was incomplete, however, and required an OS kernel. A kernel is the portion of the OS that manages communications with the computer's hardware. Linus Torvalds from Finland stepped in to fill in the

blank in the 1990s, and in 1994, he released a Unix-like OS kernel which he called Linux 1.0. More important for our purposes, he distributed his code under the terms of the GPL. It did not take long for the Linux kernel to be combined with the GNU OS to create what is today known as the GNU/Linux operating system, or simply Linux OS.

Enter Harald Welte. Welte has in the last several years contributed code to the Linux kernel, particularly with respect to network and firewall functions. He presently leads the development team for this code, which finds its home at netfilter.org.¹⁶ He claims a copyright in his contributions, which he also distributes, as he must, under the GPL.

Welte's code is present in every current release of the official Linux kernel, and many previous versions as well. Thus, any company distributing Linux software, whether standing alone or as incorporated into a hardware product (often called embedded Linux), runs the risk of hearing from Welte. Welte notes on his website that "other Linux kernel developers have transferred their rights in a fiduciary license agreement" to enable Welte and his legal team to "enforce the GPL in cases where no code originally written by . . . Welte was used."¹⁷ Presumably, this enables him to enforce the GPL against versions of the kernel written before his time as a contributor.

II.

Welte seems most concerned with two common GPL violations. First, and undoubtedly most important, he polices whether vendors make their source code available, as they are supposed to do under certain circumstances. Second, he also insists that any hardware or software distributed with the Linux kernel include a copy of the GPL and clearly state that the product contains software covered by the GPL. For those who are interested, Welte lists on his website other practices he believes violate the GPL.¹⁸

Despite the fact that in many instances a strong argument could be made for money damages in these cases, Welte has made it a practice not to ask for them.¹⁹ Welte recently stated, however, that some companies have made a "donation," as part of their settlement agreements, that "allows the respective vendor to sell already-produced products . . . during a grace period."²⁰ Whether this portends a change in strategy remains to be seen.

The relief he typically seeks, both in and out of court, is specific performance of the GPL. This is not surprising

“Welte plans to incorporate his GPL Violations project in late 2006, and it is possible that with an influx of funding he will extend his efforts to other countries within Europe and outside it, as well.”

given that Welte primarily desires that parties using GPL-covered software live up to their end of the bargain.

Welte also seeks reimbursement for the costs and attorneys' fees that he incurs in investigating and pursuing violations. The loser-pays system for fees in Germany facilitates this request. Welte undertakes most of the investigations personally, and presumably some of the costs associated with the investigations are for his time. In a recent interview, Welte estimated that the typical court case runs about 10,000 euros in fees and costs.²¹

III.

Why should U.S. companies and lawyers care about Welte? First, and most obviously, he stands ready to pursue any Linux GPL violations occurring in Germany. U.S. companies that distribute hardware and software products in Germany that incorporate Linux are therefore fair game.

Moreover, though most, if not all, of Welte's enforcement efforts have been limited to Germany, that may soon change. Welte plans to incorporate his GPL Violations project in late 2006,²² and it is possible that with an influx of funding he will extend his efforts to other countries within Europe and outside it, as well. Welte has already indicated, for example, that the project has obtained compliance agreements from companies in Korea and Taiwan.²³

Incorporating the project, and pursuing claims with the company as plaintiff, should also insulate Welte from claims for attorneys' fees in the event he loses a case. This means, in all likelihood, that he will be more likely to pursue GPL violators.

Whether Welte will begin to bring lawsuits in the United States is less clear. The rules for recovery of attorneys' fees are not as lenient in the United States as they are in Germany and in most other European countries, and this may be a sufficient deterrent. As I noted above, in Germany the winning party is generally entitled to their attorneys' fees, meaning that so long as Welte wins the case he is all but guaranteed to recover the fees and costs of the suit. Obviously, that is not always the case in the U.S., even in cases brought for copyright infringement.

Moreover, because the amount that an attorney can charge for fees is heavily regulated in Germany, Welte can, as one commentator noted, “more accurately assess his monetary risk in advance of each enforcement action.”²⁴ That luxury is not always available for legal work performed in the United States, as we all know well.

Nevertheless, Welte has said that he passes information regarding violations of the Linux OS GPL on to the Free Software Foundation's GPL Compliance Lab, here in the United States.²⁵ Though I could find no evidence of the Free Software Foundation bringing cases in the United States, that could and should soon change. It also is conceivable that the Software Freedom Law Center,²⁶ established in the United States by Eben Moglen and Daniel Ravicher, could take up U.S. cases on behalf of Welte as well.

IV.

Depending on your viewpoint, and the views of your clients, Welte may be someone to fear or someone to seek out for assistance. Regardless of your take on the matter, there can be no question that this remarkable young man is making a difference, and an early one at that, in an intersection of software development and law that has yet to fully develop.

Endnotes

1. Duke Tufty, *Open Source Software—Pandemic or Panacea?*, 6 OIPN 9 (Fall 2005).
2. Arguments that the GPL constitutes an unfair restraint on trade were recently rejected in a group of cases. See *Wallace v. IBM Corp.*, 467 F.3d 1104 (7th Cir. 2006); *Wallace v. Free Software Foundation, Inc.*, 2006 WL 2038644 (S.D. Ind. 2006); *Wallace v. Free Software Foundation, Inc.*, 2005 WL 3239208 (S.D. Ind. 2005). Another case raised the enforceability of the GPL as a matter of contract law, but the parties reached a settlement before the court decided the matter. See *Drew Technologies, Inc. v. Society of Automotive Engineers, Inc.*, No. 03-CV-74535-NGE-PJK (E.D. Mich. settled 2005). A dispute may soon be brewing in another case, however. See *Joe Brockmeier, Konsole License Violations Highlight GPL Confusion* <http://enterprise.linux.com/enterprise/06/09/29/164207.shtml?tid=41> (Oct. 6, 2006).

3. *Welte v. Sitecom GmbH*, No. 21 O 6123/04 (Dist. Ct. Munich May 19, 2004), available at http://www.jbb.de/judgment_dc_munich_gpl.pdf.
4. *Id.*
5. Ingrid Marson, *Fortinet Settles GPL Violation Suit* http://news.com.com/2100-7344_3-5684880.html (Apr. 26, 2005).
6. *Id.*
7. See e.g., Thomas Hoeren, *The First-Ever Ruling on the Legal Validity of the GPL—A Critique of the Case*, http://www.oii.ox.ac.uk/resources/feedback/OIIFB_GPL3_20040903.pdf (Sept. 3, 2004).
8. *Welte v. D-Link Germany GmbH*, No. 2-6 O 224/06 (Dist. Ct. Frankfurt Sept. 6, 2006), http://www.jbb.de/urteil_lg_frankfurt_gpl.pdf; see also Mayank Sharma, *GPL Passes Acid Test in German Court* <http://www.linux.com/article.pl?sid=06/09/24/1252212> (Sept. 24, 2006).
9. Gpl-violations.org Project Prevails in Court Case on GPL Violation by D-Link, http://www.gpl-violations.org/news/20060922-dlink-judgement_frankfurt.html (Sept. 22, 2006).
10. See <http://www.jbb.de>.
11. Its web presence can be found at <http://www.gpl-violations.org>. Welte's efforts to police GPL violations are also recounted on his personal blog at <http://gnumonks.org/~laforge/weblog/linux/gpl-violations/index.html>.
12. See <http://www.gpl-violations.org/about.html#whois>.
13. Harald Welte on the Flood of GPL Violations, <http://lwn.net/Articles/186944> (June 19, 2006).
14. Ingrid Marson, *Defender of the Linux Faith*, http://news.com.com/Defender+of+the+Linux+faith/2100-7344_3-5625667.html (Mar. 18, 2005).
15. <http://www.gnu.org/licenses/gpl.html>.
16. <http://www.netfilter.org>.
17. <http://gpl-violations.org/about.html>.
18. <http://www.gpl-violations.org/faq/vendor-faq.html>.
19. Harald Welte on the Flood of GPL Violations, *supra* n. 13.
20. *Id.*
21. Marson, *supra* n. 14.
22. <http://www.gpl-violations.org/faq/violation-faq.html>.
23. Harald Welte on the Flood of GPL Violations, *supra* n. 13.
24. Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 Berkeley Tech. L.J. 443, 471 n. 157 (2005).
25. *Id.*
26. <http://www.softwarefreedom.org>; see also Joe Brockmeier, *Enforcing the GPL*, <http://software.newsforge.com/article.pl?sid=06/05/01/1938223&from=rss> (May 11, 2006).

Semi-Open Source: The Case for Hybrid Licensing



By William Glasson
Mentor Graphics Corporation
will_glasson@mentor.com

1. Introduction

Despite the increasing popularity of open source software like Linux and Mozilla, many proprietary software developers remain wary of open source code and the “viral” effects of its most popular license, the GNU General Public License (GPL).¹ This wariness commonly stems from uncertainty about the bargain struck with OSS licensors when building a revenue model around “free” code.² For many of these developers, the GPL's signature terms are anathema to how they profit from their intellectual property.³

Some software companies are using an alternate development and licensing model to address this concern. Sometimes called a “hybrid” or “dual license” model, these companies release code under the GPL and under a proprietary license.⁴ This article will examine hybrid licensing and assess when a software purveyor could successfully implement this model.

2. Hybrid Licensing In a Nutshell

Hybrid licensing requires that the developer own the licensed code. This licensing model may also require a developer to change the way it develops, as well as licenses, its products. Under a hybrid license model, a developer designates code that will be licensed under an open source license (commonly the GPL) and code that will be available under a proprietary license.⁵ Recipients may elect to obtain the software under the open source license or the proprietary license. The open source code

is often more limited than the proprietary code and subject to a re-licensing obligation on the same terms.

One example is Red Hat, a leading Linux distributor. Red Hat uses a kind of “bundling” hybrid model.⁶ Under its model, licensees may receive a standard Linux distribution or may pay an additional fee for a high-end version containing additional proprietary programs that are not available under the GPL.⁷ The high-end version’s additional cost is justified by these value-added components, which may incorporate proprietary intellectual property unavailable in the GPL distribution.

Red Hat’s high-end version illustrates one of a variety of hybrid models currently in use. A second popular hybrid model, “concurrent” hybrid licensing, involves allowing recipients to elect to license the code under a proprietary license and an open source license.⁸ Although apparently counterintuitive, some developers may find that this model offers the only feasible way to foster widespread commercial exploitation of their code.⁹ The developers most likely to benefit from this model are those committed to maintaining an open source implementation because their products are incorporated into a licensee’s product and distributed.

Developers adopting this model must either require that all contributors assign their copyright interest in contributed code to the copyright holder or project administrator or address what hackers call the “forking problem.”¹⁰ Forking arises when a code platform splits and evolves in different, incompatible, directions.¹¹ Because a developer must surrender control over code released under the GPL and other common open source licenses, and allow other subsequent developers to evolve the code as they choose, the code often diverges from the original developer’s own proprietary platform. Accordingly, a developer employing this hybrid model may want to track how the open source code evolves and ensure that its proprietary version contains features found in the open source implementation that its users demand. However, depending on how the code evolves, the original programmers may be independently required to develop functionality that has been built into the GPL version. If subject to the GPL, the developer must ensure these features are developed without the aid of the GPL code, because the GPL assimilates code that is derived from GPL-licensed code.¹²

As noted above, implementing a hybrid model requires that developers thoughtfully map each code platform. For developers employing a model similar to Red Hat’s, the open source product’s architecture must allow developers to incorporate additional components

easily. As mentioned above, a developer may need to ensure that its components licensed under a proprietary license are not derived from the open source code.¹³ Also, the developer’s product must be modular and not contain any code based on a “work” licensed under the GPL.¹⁴ To the extent that code licensed under the GPL is distributed in combination with proprietary code, the code should remain “modular” such that “identifiable sections of [the proprietary] work are not derived from the Program, and can be reasonably considered independent and separate works in themselves.”¹⁵ Code developed in this manner is less likely to be subject to the GPL’s “viral” licensing provision.¹⁶

Alternately, where developers offer an open source and a commercial version, as in the concurrent hybrid model, licensees would prefer to be able to switch out the open source version for the commercial version with little additional engineering work. In most cases, programmers use standard or well-published interfaces to accommodate this requirement.¹⁷

3. When is Hybrid Licensing Appropriate?

Not all software products or implementations can be successfully licensed using a hybrid model. Apart from the engineering requirements noted in this article, these products must also appeal to open source and proprietary licensees—groups that ultimately use and distribute the code. Appealing to these two user groups can often involve radically different approaches.¹⁸ Users drawn to open source implementations are generally very computer literate, caring more about whether a product is sophisticated or challenging than whether it is easy to use.¹⁹ By contrast, many commercial customers – the majority of computer users – tend to value ease of use, reliability, and core functionality over a product’s technical sophistication.²⁰

Developers that can appeal to the gamut of these interests may benefit from a hybrid model.²¹ Alternately, developers whose main criteria are price and source code access may also benefit from a hybrid model.²² Under either analysis, however, the success of a hybrid model is also contingent on whether the model and the product are complementary.

Accordingly, with notable exceptions, many products released under a hybrid model are niche products that appeal to a narrow user base.²³ Typically, these are standout products relative to their markets and are used by sophisticated customers. Consistent with the models identified above, most hybrid products are incorporated

into other products and/or serve as platform technologies, such as an operating system or database management application.²⁴ These products can be bundled with hardware or software, or can be independently available.²⁵ Thus, if the customer-base supports a hybrid product, the last dispositive inquiry is whether the product's open source component will be compatible with an open source, or "peer-produced," development model.²⁶

3.1 When Will an Open Source Model Work?

Because hybrid licensing involves an open source implementation, a developer must consider whether a code base or executable model is suitable for open source development and licensing. The first requirement is a program or cohesive code platform. While some open source products arguably have germinated from less, few projects succeed unless they begin as at least a semi-operational product.²⁷ Accordingly, releasing projects prior to their beta stage of development may limit the interest expressed by community users.

Next, a developer must insure that its project is one that "hackers" will use, modify, discuss, and disseminate; or, that it will work in the "bazaar."²⁸ Characteristically, these projects are "fluid, user-driven," continuous, incremental, and managed by a central figure that is well respected in the developer community.²⁹ Additionally, these developers generally respond more favorably to projects that are useful to individual as well as enterprise users.

Developers can improve a project's chances for success by ensuring that its code platform is divisible.³⁰ There are three primary attributes that promote divisibility: (i) the code platform is modular; (ii) the platform contains granular components; and (iii) the components can be integrated at minimal cost.³¹ To Yochai Benkler, a modular code platform can be divided into encapsulated modules that are developed independently from the larger platform.³² A platform contains granular components if the module sizes are small enough to ensure that multiple, minimally skilled individuals will be sufficiently motivated to work on those individual modules.³³ Finally, this production scheme works only if the modules can be integrated easily, and the work products vetted to ensure high quality contributions.³⁴ In most instances, the economic feasibility of integrating component modules turns on the availability of complementary technologies, such as concurrent versioning systems (CVS), which software programmers use to track code revisions.³⁵

4. Benefits of a Hybrid Model

Developers whose products are compatible with a hybrid model can often benefit economically or administratively from a hybrid licensing and development scheme. Below are a handful of examples illustrating developers that economically or administratively benefited from hybrid models.

4.1 Economic Benefits

Hybrid licensing can lower development costs, increase technical support revenue, and create new markets for a developer's products.³⁶ A developer may be able to lower development costs by segmenting products into proprietary and open source modules, and by focusing development efforts on the proprietary module.³⁷ Alternately, a developer with a clear product roadmap and an established core of open source developers can reduce its testing budget by licensing a platform beta product under the GPL and adapting its corresponding proprietary module to the GPL component as it is refined.³⁸ Additionally, open source developers that have historically derived their revenue only from non-licensing-based activities can employ a hybrid model to expand their revenue streams.

For example, in December 2002, Mandriva instituted a type of dual licensing model to expand its revenue opportunities.³⁹ The change came after an unprofitable four years, despite the popularity and widespread use of the company's Linux operating system.⁴⁰ Prior to moving to a hybrid model, Mandriva licensed its products under the GPL and derived its revenues from service and contract sales.⁴¹ The company switched to a hybrid model and returned to profitability within a year.⁴² Switching to a hybrid model enabled Mandriva to leverage its installed customer base, and similarly to Red Hat, charge license fees for add-ons sold as part of its high-end Linux product.⁴³ Further, the company was able to realize this additional revenue without a significant decrease in service and consulting revenues.

4.2 Administrative Benefits

Hybrid licensing can solve administrative challenges and increase a developer's revenue opportunities. The benefits can be myriad. First, code licensed under open source licenses with re-licensing requirements, such as the GPL, facilitates wider proliferation of the code and a public feedback loop, allowing developers to respond to user issues more quickly – either with open source or proprietary solutions.⁴⁴ Second, developers can more easily cease supporting old platforms if they release

them under an open source license. Both developers and users benefit by allowing users who rely on older platforms to extend them on their own. By ensuring that customers retain access to these products, developers can often maintain these customer relationships, which may provide the developer with future revenue opportunities for proprietary products. Last, because open source code is typically more reliable than proprietary code, the developer can concentrate development efforts on the complementary proprietary products, which similarly improves the quality of these products.⁴⁵

For example, Artifex software uses a dual licensing model to market its various script interpreters.⁴⁶ These commercial interpreters provide licensees (mainly other software developers) with a proprietary alternative to GPL interpreters such as GNU Ghostscript.⁴⁷ Artifex benefits from this complementary arrangement in a number of ways. Because the GPL interpreters are so specific and widely used, the company is spared some of the burden of developing a product roadmap or gauging customer demand for new functionality. Instead, Artifex can respond to new customer demand either by monitoring programmer listserves (where programmers implementing the GPL analog post comments and questions), or through direct customer feedback. This dynamic is effective and reliable because customers interested in Artifex's interpreters typically integrate the GPL version initially, while engineering their products, and then contact Artifex when the product nears beta stage. Further, because of this feedback loop and the technically arduous nature of writing such interpreters, few offer a competitive product or engineer their own solution.⁴⁸

This hybrid model also benefits Artifex by reducing the demand for basic technical support. All Artifex licensees are required to purchase an initial support contract as part of the commercial license. However, due to the extensive dialogue on GPL interpreter listserves and other electronic forums, open source programmers can receive answers to their technical support questions by posting to these sites. Accordingly, these forums minimize the number of support incidents lodged with Artifex under its mandatory support program, allowing Artifex to realize higher profit margins. Customers also benefit from this arrangement by, theoretically, receiving higher quality support when they do access Artifex support resources.

5. Conclusion

Hybrid licensing, in which code platforms are released under an open source as well as a commercial license, offers some software developers an attractive compromise between a "copyleft" policy, and the exclusivity of proprietary licensing. For developers with compatible products and customers, hybrid models can expand revenue opportunities, reduce administrative burdens, and possibly create new markets.

Endnotes

1. Kem McClelland, *A Practical Guide to Using Open Source Software in a Time of Legal Uncertainty*, 743 PLI/Pat 351, 353-66 (2003). "GNU" is a recursive acronym that stands for "GNU's Not UNIX." [from <http://www.gnu.org/>].
2. Ieuan G. Mahony & Edward J. Naughton, *Open Source Software Monetized: Out of the Bazaar and into Big Business*, 21 *The Computer & Internet Lawyer* 1 (Oct. 2004). The GPL was created by the Free Software Foundation, which prefers the term "free" software to the better known term, "open" software (derived from "open" source), which is the term the Open Software Initiative uses.
3. The GPL's signature terms are: (i) the freedom to run the program without restriction; (ii) the freedom to learn how the program works and to modify it (i.e., to have access to its source code); and (iii) the freedom to distribute the modified or unmodified program – in any form – to the public. [from <http://www.gnu.org/>].
4. *Id.* at 4-5. This model marginalizes the "viral" aspect of the GPL and similar open source licenses, and enables groups such as Red Hat and IBM to offer proprietary and open source code. IBM refers to this model as the "proprietary with open platform bundling" model. For the purposes of this paper, however, I will focus on same-application code offerings (i.e., proprietary and GPL code used simultaneously in the same application or niche). "Hybrid" licensing is also sometimes called "dual" or "integrated" licensing.
5. The Open Software Initiative publishes the official list of licenses that comply with their definition (recognized as the official definition) of "open source." Open Source Initiative, Open Source Initiative OSI – Licensing, <http://www.opensource.org/licenses> (accessed Nov. 14, 2006).
6. *Going Hybrid: Rumours of Open-source Software's Demise are Exaggerated*, *The Economist*, http://www.economist.com/printedition/displayStory.cfm?Story_ID=1251254 (accessed Nov. 14, 2006) ; Stacey Quandt, *Taking Linux to the Bank*, *Linux Magazine*, <http://www.linux-mag.com/content/view/1733/2068/1/0/> (Sep. 15, 2004).
7. Quandt, *supra* n. 5.
8. This is closer to a more traditional "dual" licensing model.
9. GNU Project - Free Software Foundation, *GNU General Public License, version 2* (June 1991), <http://www.gnu.org/licenses/gpl.html>, (updated Oct. 8, 2006) [herein-after GPL].

10. Eric S. Raymond, *Homesteading the Noosphere*, <http://catb.org/esr/writings/cathedral-bazaar/homesteading/> (updated Aug. 24, 2000).
11. Glyn Moody, *Rebel Code: Linux and the Open Source Revolution 171* (Perseus Books Group 2001).
12. GPL, *supra* n. 8. This type of development is known as "closed-room" engineering.
13. *Id.* Yochai Benkler, *Coase's Penguin, or, Linux and the Nature of the Firm*, 112 Yale L.J. 369, 378 (2002).
14. Peter Brown, *Legal Issues in the Open Source Community*, 780 PLI/Pat 309, 319-20 (2004).
15. David W. Opperbeck, *The Penguin's Genome, or Coase and Open Source Biotechnology*, 18 Harv. J. L. & Tech. 167, 198 n. 170 (2004).
16. Mahony, *supra* n. 2.
17. GPL, *supra* n. 8; see also *Progress Software Corp. v. MySQL AB*, 195 F. Supp. 2d 328, 329 (D. Mass. 2002).
18. Knowledge@Wharton, *Getting the Most Value out of Open Source Software*, <http://knowledge.wharton.upenn.edu/index.cfm?fa=viewArticle&ID=946> (Mar. 25, 2004); LWN.net, *On the Dual-License Model*, <http://lwn.net/Articles/172128/> (Feb. 15, 2006).
19. Raymond, *supra* n. 9; see also Moody, *supra* n. 10, at 29.
20. Raymond, *supra* n. 9; see also Moody, *supra* n. 10.
21. See Stephen Shankland, *Profit helps Mandrake out of bankruptcy protection*, <http://news.com.com/2100-7344-5181884.html?tag=nl> (Mar. 30, 2004).
22. Martin Fink, *The Business and Economics of Linux and Open Source* 177 (Prentice Hall PTR 2002).
23. This statement is based on a unit sales figure. Although server and networking equipment sales are likely more economically significant, because these are high-end products, maintained by a comparatively few individuals, the unit sales figure better represents the actual reach of open source products.
24. See e.g. Wikipedia, *Comparison of Linux Distributions*, http://en.wikipedia.org/wiki/Comparison_of_Linux_distributions (last accessed Nov. 14, 2006); MySQL AB, *MySQL Products*, <http://mysql.com/products/> (last accessed Nov. 14, 2006); Oracle, *Oracle and Sleepycat*, <http://www.oracle.com/sleepycat/index.html> (last accessed Nov. 14, 2006); see also KGDB: Linux Kernel Source Level Debugger, *About KGDB*, <http://kgdb.linsys-soft.com/about.htm> (last accessed Nov. 14, 2006).
25. Knowledge@Wharton, *Open Source: Closing, Closing ...*, <http://knowledge.wharton.upenn.edu/index.cfm?fa=viewArticle&ID=827> (Aug. 13, 2003). IBM, Hewlett-Packard, and Dell all bundle Linux with their servers. Novell now embeds complementary proprietary applications with an open source platform, a Linux variant.
26. Eben Moglen, *Anarchism Triumphant: Free Software and the Death of Copyright*, http://www.firstmonday.dk/issues/issue4_8/moglen/index.html (Aug. 2, 1999).
27. See Lewis Carroll, *Alice's Adventures in Wonderland*, <http://www.literature.org/authors/carroll-lewis/alices-adventures-in-wonderland/chapter-12.html> (updated May 23, 2005): "'Begin at the beginning,' the King said gravely"
28. See e.g. Moody, *supra* n. 10, at 25-28.
29. Benkler, *supra* n. 12, at 376; Eric S. Raymond, *The Cathedral and the Bazaar*, http://www.firstmonday.dk/issues/issue3_3/raymond/ (Mar. 2, 1998). Raymond's use of "bazaar" is romantic flair and an attempt to invoke the rich markets of antiquity where communities collected to interact as well as sell their homespun wares.
30. Raymond, *supra* n. 28.
31. Benkler, *supra* n. 12, at 378.
32. *Id.* at 378-79.
33. *Id.*
34. *Id.*
35. *Id.*
36. *Id.*
37. Mahony, *supra* n. 2; see also Charles H. Cella & Edward J. Kelly, *Considerations for Companies Developing Software Under the Open Source Model*, 4 Cyberspace Lawyer (Sept. 1999).
38. This assumes there's a robust and committed open source development community working on the open source module.
39. See, e.g., Moody, *supra* n. 10, at 55-86. The early Linux principals are an example of a core development group.
40. Heather Meeker, *Dual-Licensing Open Source Business Model*, <http://linux.sys-con.com/read/49061.htm> (Apr. 6, 2005); see generally *Mandriva Linux, About Us*, <http://www.mandriva.com/company/about> (last accessed Nov. 14, 2006).
41. Shankland, *supra* n. 20.
42. Mandriva Linux, *Legal FAQ*, http://www.mandriva.com/en/company/legal_faq (last accessed Nov. 14, 2006); see generally <http://www.mandriva.com/en/company/about> (last accessed Nov. 14, 2006).
43. Shankland, *supra* n. 20.
44. Meeker, *supra* n. 39.
45. *Going Hybrid*, *supra* n. 5.
46. *Id.*
47. See also Artifex Software Inc., *About Artifex*, <http://www.artifex.com/aboutartifex/index.htm> (last accessed Nov. 14, 2006). Artifex offers many of the only commercial interpreters for the PostScript, Ghostscript, PDF, and PCL formats and page description languages.
48. See GNU Project – Free Software Foundation, *GNU Ghostscript*, <http://www.gnu.org/software/ghostscript/ghostscript.html> (updated Aug. 19, 2006).
49. John Koenig, *Seven open source business strategies for competitive advantage*, <http://www.itmanagersjournal.com/feature/314> (May 13, 2004). This article covers many of the advantages described in this paragraph.

Trademark Enforcement for National Brands Made Easier – Not So For Others: Significant Changes to Trademark Dilution Act Signed into Law



By Sheila Fox Morrison

Associate, Davis Wright Tremaine LLP
(503) 778-5311
sheilamorrison@dwt.com

On Oct. 6, 2006, President Bush signed into law the Trademark Dilution Revision Act of 2006. The Act modifies the Federal Trademark Dilution Act (FTDA) making it easier to prove trademark dilution for national brands, but eliminating dilution as a basis for relief for niche market and regionally famous marks. The Act is in response to a recent Supreme Court case and various conflicting decisions of the federal courts of appeals. With this amendment, proof that a defendant's activities are likely to dilute a mark now is sufficient to obtain an injunction, and various other issues surrounding dilution law have been clarified.

Trademark dilution occurs when the capacity of a famous trademark to identify and distinguish goods or services is weakened, even if there is no competition or likelihood of confusion. For example, use of TIFFANY in connection with restaurant services, or use of POLO for an adult entertainment business.

This much-anticipated revision of the FTDA is primarily a response to the Supreme Court's decision in *Moseley v. V Secret Catalogue, Inc.* 537 U.S. 418 (2003), in which the Court held that actual dilution, and not merely a likelihood of dilution, had to be proved to obtain an injunction against dilution of a mark. Congress responded to *Moseley* by enacting the Trademark Dilution Revision Act to explicitly allow an injunction when a likelihood of dilution is demonstrated. Actual dilution no longer need be proved.

Congress also addressed numerous conflicts among the federal courts of appeals that have muddied trademark dilution law. The amendment makes clear that dilution protection extends to trade dress (the distinctive appearance of a product or its packaging), and prohibits activities in which a famous mark is "tarnished," such as when it is used with a disreputable product or service. In addi-



By Vanessa Usui

Associate, Davis Wright Tremaine LLP
(503) 778-5279
vanessausui@dwt.com

tion, what constitutes a famous mark that qualifies for dilution protection is now defined as one that is "widely recognized by the general consuming public of the United States." This clarification rejects the holdings of some courts that a mark can be famous in a niche market or a regional market only. A separate fair-use exemption protects parody, comment and criticism while the FTDA's preemption of state dilution law has been expanded.

Text of the Trademark Dilution Revision Act of 2006 can be found at: <http://www.govtrack.us/data/us/bills.text/109/h/h683.pdf>.

For more information, please contact:

Sheila Fox Morrison
Portland, Oregon
(503) 778-5311
sheilamorrison@dwt.com

Vanessa Usui
Portland, Oregon
(503) 778-5279
vanessausui@dwt.com

Oregon State Bar
Intellectual Property Section
5200 SW Meadows Road
PO Box 1689
Lake Oswego, OR 97035-0889

Presorted Standard
US Postage
PAID
Portland, OR
Permit No. 341
