301 Understanding Open Source Software

Eben Moglen
Chairman
Software Freedom Law Center

Matthew A. Neco
Vice President & General Counsel
Stirling Bridge, Inc./Morpheus

Mark H. Webbink
Executive Vice President and General Counsel
Red Hat, Inc.

Faculty Biographies

Eben Moglen

Eben Moglen is professor of law at Columbia Law School in New York and founding director-counsel of the Software Freedom Law Center. He served as law clerk to Judge Edward Weinfeld of the Southern District of New York and Justice Thurgood Marshall.

Before and during law school he was a designer and implementer of advanced computer programming languages at IBM's Santa Teresa Laboratory and Thomas J. Watson Research Center. Since 1993 he has served pro bono publico as general counsel of the Free Software Foundation.

He was awarded the Electronic Frontier Foundation's 2003 Pioneer Award for contribution to freedom in the electronic society.

His JD and PhD in history were earned during what he sometimes refers to as his long dark period in New Haven.

Matthew A. Neco

Matthew A. Neco is vice president and general counsel for Los Angeles based Stirling Bridge, Inc., and its subsidiaries, including most notoriously StreamCast Networks, Inc., dba Morpheus, the decentralized public peer-to-peer file search and sharing software application developer and distributor.

Before joining Stirling Bridge, Mr. Neco worked for several privately held entertainment, media, and technology entities, including one he was co-founder of, and he operated a law and mediation office in West Los Angeles. He has focused on various aspects of the entertainment industry in film and TV (representing mostly nascent actors, writers, independent producers), the music industry (representing mostly nascent artists, songwriters, producers, and independent labels), and the internet and new media, as well as reorganizations, work-outs, litigation, and asset acquisitions in the bankruptcy courts.

Mr. Neco received his B.A. from The University at Albany (State University of New York).
He received his J.D. from the University of Wisconsin Law School in Madison, Wisconsin.

Mark H. Webbink

Mark H. Webbink is the senior vice president, deputy general counsel and assistant secretary for Red Hat, Inc. in Raleigh, North Carolina. His responsibilities include intellectual property, licensing, litigation, and public policy.

Prior to assuming his present role, Mr. Webbink served as Red Hat general counsel and secretary. Prior to joining Red Hat, he was associated with Moore & Van Allen, PLLC in Durham, North Carolina, where his practice focused on intellectual property transactions and general corporate work.

Mr. Webbink is a member of ACC, ABA, and North Carolina Bar Association. He presently serves as chairman of the board of directors of the Software and Information Industry Association. He is also active in the European committee for Interoperable Systems, ACC, International Trademark Association, and Licensing Executives Society.

Mr. Webbink received a B.A. from Purdue University, an M.P.A. from the University of North Carolina-Chapel Hill, and his J.D. from North Carolina Central University School of Law.

Free and Open Source Software: Risks Your Company's Software Developers Might Not Be Aware Of[1]

By Matthew A. Neco and Wendy Millar Goodkin[2]

Depending on the size of the company and legal department you work for as in-house counsel you might think you are aware of almost every contract and licensing agreement that your company has become a party to since you started with the company. You might think that you or some other lawyer has reviewed and advised on every agreement. But if your client is a software development company or a company that has an information technology department that modifies or tweaks software your company owns (that is, that your company developed) or licenses as a licensee, you might be in for a little surprise. The fact of the matter is that there may be licensing agreements that have been entered into on behalf of your client company without the legal department even knowing it. No, it's not that your CEO and business development teams are signing contracts before a lawyer reviewed and advised. (Well, maybe they are.) But there are threats you might not have thought about. They are your company's very independent thinking software developers, engineers, and IT staff. Engineers are logical thinkers, so if you take the time to explain it right they'll get it (and notwithstanding that some of us in-house lawyers sometimes think it might seem easier to get a hungry Pit Bull not to attack a pound of raw meat then it is to get engineers to pay attention to lawyers, usually they don't want to do anything that will hurt the company). Unless you've got them well educated and sensitive to certain issues involving FOSS they may be posing silent threats to your client, entering into software licensing agreements in a willy nilly fashion.

You ask, Just what kind of agreements? Agreements governing software licensed as Free and Open Source Software ("FOSS"). And this may have profound implications for proprietary software owned by you company or which your company may be licensee to. Your first thought might be that there can't be much harm in software that's "Free" and "Open," other than thinking You get what you pay for: If it's free how good can it be? You think you'll leave it up to the engineers and IT staff to deal with all the bugs, and other program problems that must be in this free software. Otherwise, you think, it should be safe for your company to use FOSS. Wrong. Unlike software that has truly been released into the public domain, FOSS software is typically subject to licensing agreements. Licensing agreements that might require your company to perform certain obligations. Licensing agreements that may, inadvertently or otherwise, expose your company's proprietary software (or software that it has licensed as licensee) to what is known as spoilation (spoliation in the litigation world) or pollution.

What are the possible results of spoilation or pollution? Well, if your client has developed and owns proprietary software your company might be required to disclose or publish some or all of its proprietary code. Your client might be required to make its software available to the world for free under the same license that governed the FOSS used by the engineers. There go the trade secrets. There goes the software product your company sells. There goes your job.

With this huge potential risk in mind, you may be thinking that the best advice would be to tell your company that it shouldn't ever use FOSS. This advice just might be bad advice too. FOSS can offer your company many business advantages. First, FOSS is economical in that your company will not need to spend money on licensing another company's proprietary software or spend valuable time and resources developing and testing its own software, or components for software that it owns or is licensee of. Second, FOSS is readily available. All you need to do is conduct a search using your favorite search engine or Morpheus™ [3] to see how easy it is to find and download FOSS. In addition, unlike developing your own software, FOSS may have been read, tested, tried, redistributed, improved, and modified by many software developers, thus, resulting in higher quality software than your company could produce on its own.[4] Lastly, FOSS is flexible. Most FOSS licenses allow your company to modify the code to fit its business needs. Although your company must be careful as to what is required of it when such a derivative work is made from modifying the software, the benefits are something that may

[1] The article and the contents herein are meant to provide the reader with general information. Nothing herein shall be construed as legal advice.

[2] Matt is Vice President & General Counsel and Wendy is Staff Attorney of Stirling Bridge, Inc. and its subsidiaries including StreamCast Networks, Inc. (dba Morpheus™). The opinions, viewpoints, warnings, etc., expressed herein are solely those of Matt and Wendy, and are not those of StreamCast (except, perhaps at Footnote 3), or any other client of Matt or Wendy.

[3] Yes, that's a shameless little plug. And here's a warning: Do NOT use Morpheus™ to download content that is not authorized by the rights holders for free downloading, or is not otherwise authorized for downloading. Copyright and patent infringement are not condoned. This article is not intended to actively induce copyright infringement or the violation of <u>any</u> law or the rights of any person or entity.

[4] http://www.opensource.org/

not typically be available when you are a licensee of proprietary software either because it would be a violation of the licensing agreement or because proprietary software only includes the object code (as compared to FOSS which includes both the object code and the source code[5]) which does not provide the information necessary for a developer to alter or modify the software.

So now that you know that your Company may be using FOSS and you can't simply ignore it, or have it go away, what can you do? Well, it's time to get educated and get talking. Visit http://www.opensource.org/ and review the numerous FOSS licenses that are available. Remember not all FOSS licenses are the same and the differences can be significant for your company. For example, Section 2 of the General Public License ("GPL") states:

> You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
>
> a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b) 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.
>
> c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)
>
> These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
>
> Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
>
> In addition, ***mere aggregation*** of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License. (Emphasis added.)[6]

Therefore, integrating your company's software with FOSS licensed under the General Public License (GPL) may pollute or spoil your company's proprietary software requiring your company to disclose or publish some or all of its proprietary code.

After you are familiar with the various licenses, you should talk to your company's developers. You should know what software the developers would like to use, what licensing agreement(s) may be associated with that software, and how the developer would like to use that software – including how modular, or tightly "integrated" or intertwined the FOSS is likely to be with the proprietary software. What constitutes "*mere aggregation*" (see bold italics, above) versus this idea of integration is a legal decision, and there's not a lot of precedential case law out there. Yet. And you don't want to be a test case. However, to help you and the developers to make this determination the Free Software Foundation ("FSF") has suggested that you look at the mechanism of communication, including whether the modules are included in the same executable file (suggesting integration), whether the modules are designed to run linked together in a shared address space (suggesting integration), or whether there are pipes, sockets and/or command-line arguments used to communicate between the proprietary software and FOSS (suggesting "*mere aggregation*"). In addition, you will also need to discuss with the developers what type of information is being exchanged. For example, if there is an exchange of complex internal data structures this could be a basis for showing integration.[7]

---

[5] Open Source Software must include both object code and source code to be considered open source software, although the source code may be available by well publicized means rather than disclosed with the FOSS itself, see http://www.opensource.org/docs/definition.php and http://www.gnu.org/philosophy/free-sw.html.

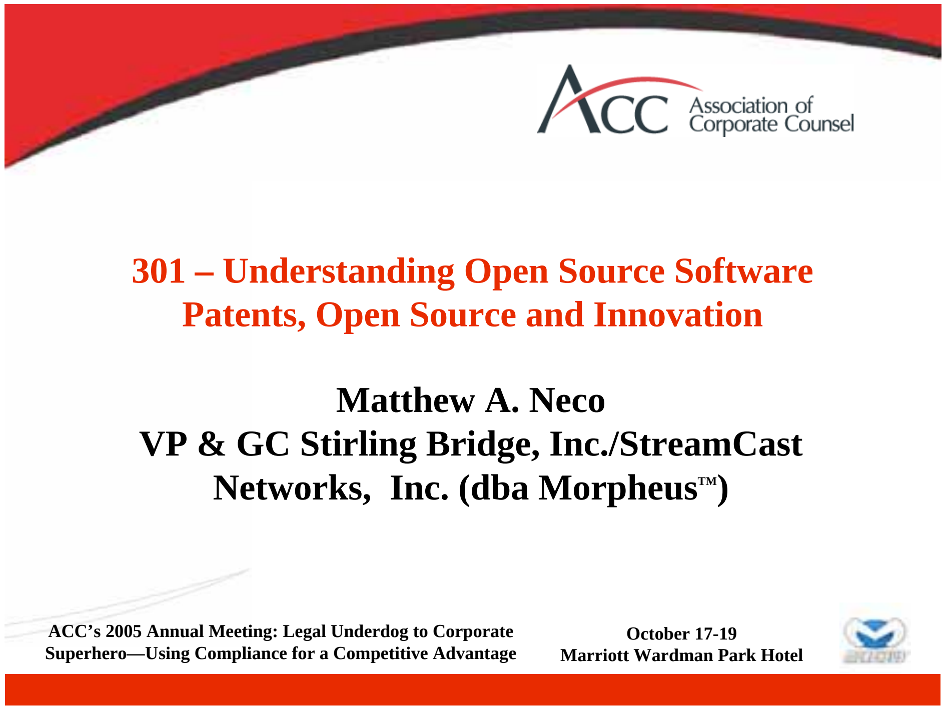
[6] The GNU General Public License (GPL), Version 2, June 1991, Copyright (C) 1989, 1991 Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA (See http://www.opensource.org/licenses/gpl-license.php for GPL License)

[7] See, http://www.gnu.org/licenses/gpl-faq.html

Once you have that information, you then have the knowledge to advise you company, from a legal perspective, whether FOSS is right for it, or advise utilizing the FOSS in a more modular – merely aggregated - manner, rather than tightly intertwined or integrated, so as to attempt to minimize the risks of spoilation. If the business decision is to use FOSS, then you will need to advise regarding how to comply with the licensing agreement, how to use FOSS in conjunction with the company's proprietary software to avoid spoilation. However, it is not just your client's proprietary software that you need to be concerned about. If your client is the licensee of someone else's software that might be proprietary in whole or in part your client might be violating the licenses by incorporating FOSS with the licensed proprietary software. In addition, if your client is already the licensee of FOSS, your need to review the license to ensure that any future FOSS licenses will not be incompatible with the first.

And don't forget, you can't just talk to the developers and walk away. You must stay involved. If you don't stay involved the developer may have already included open source software thinking that he/she understood the implications. In addition, new versions of the FOSS may be released under a different licensing agreement, thus, requiring new and different compliance requirements.

# # #



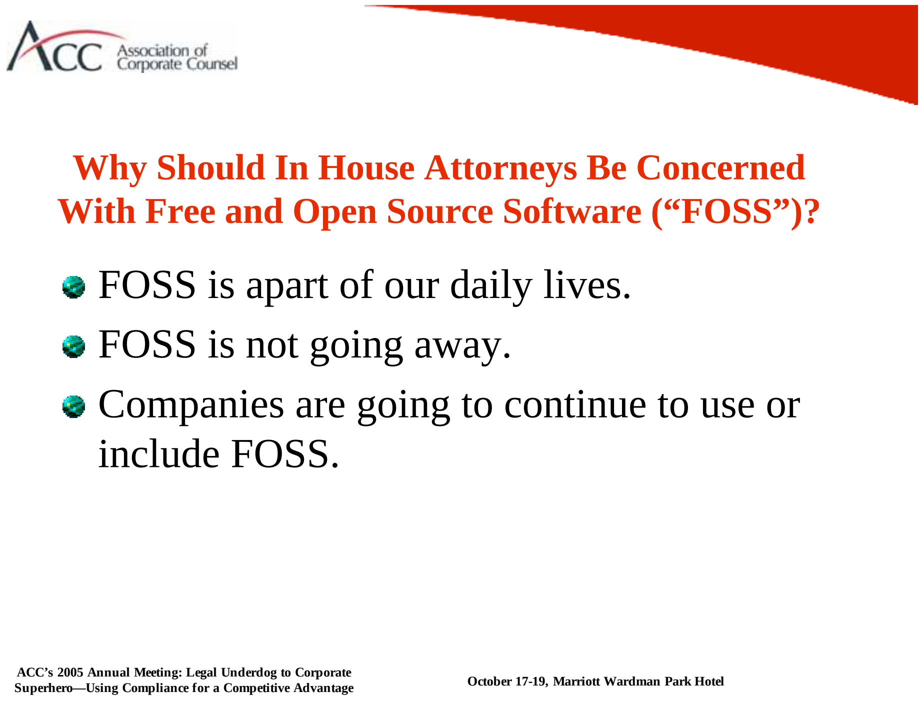
ACC Association of Corporate Counsel

301 – Understanding Open Source Software Patents, Open Source and Innovation

Matthew A. Neco
VP & GC Stirling Bridge, Inc./StreamCast Networks, Inc. (dba Morpheus™)

ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage

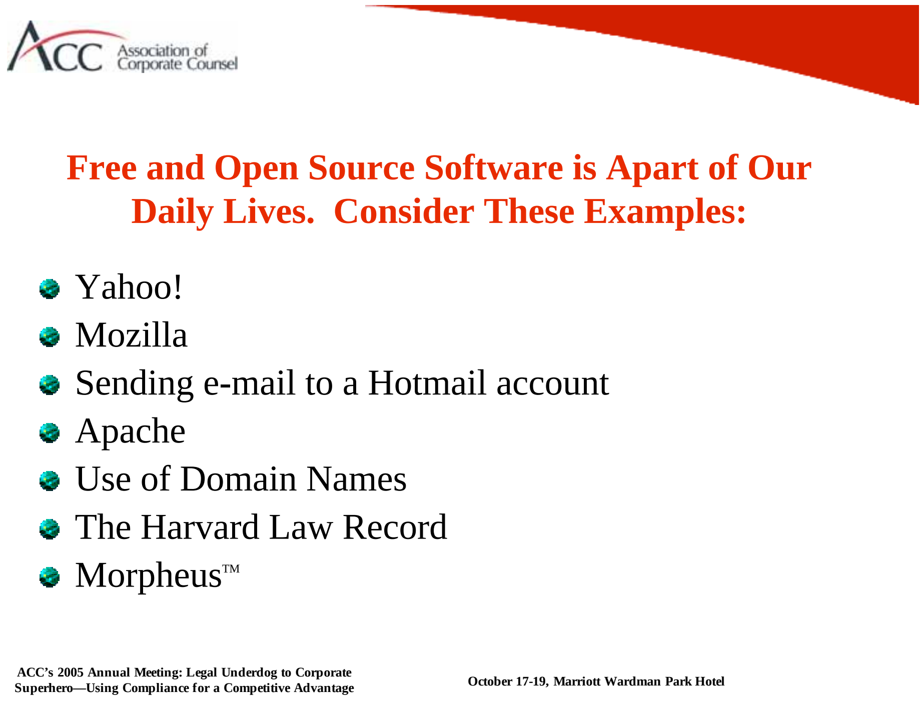
October 17-19
Marriott Wardman Park Hotel



Why Should In House Attorneys Be Concerned With Free and Open Source Software ("FOSS")?

- FOSS is apart of our daily lives.
- FOSS is not going away.
- Companies are going to continue to use or include FOSS.

ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage October 17-19, Marriott Wardman Park Hotel



Free and Open Source Software is Apart of Our Daily Lives. Consider These Examples:

- Yahoo!
- Mozilla
- Sending e-mail to a Hotmail account
- Apache
- Use of Domain Names
- The Harvard Law Record
- Morpheus™

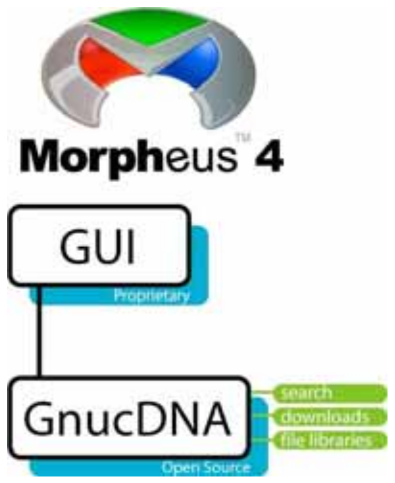
ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage October 17-19, Marriott Wardman Park Hotel

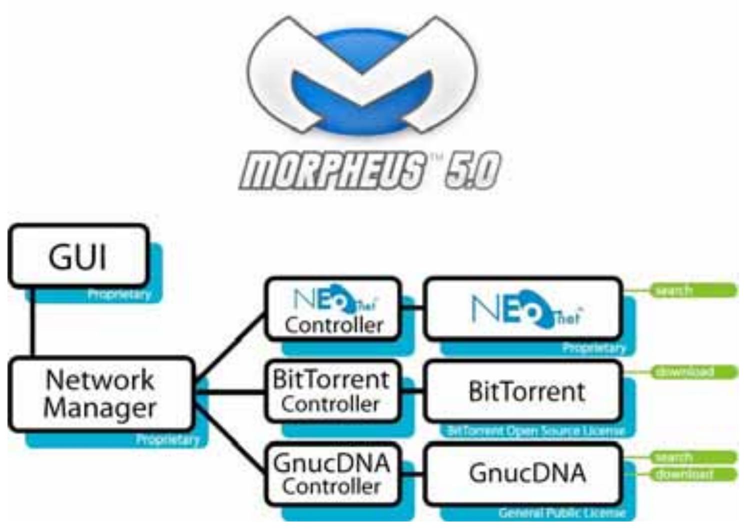
Free and Open Source Software is Not Going Away

- Developers like the prestige/reputation associated with developing FOSS.
- Companies can create both a FOSS version and an enhanced or superior proprietary version of their software, and use the FOSS version as a form of promotion for the proprietary version.
- Companies can create a business around providing technical support for their FOSS.
- Companies can create FOSS in order to get peer review of the code.
- Companies might create FOSS in order to gain good will.

Companies Are Going to Continue to Use or Include Free and Open Source Software

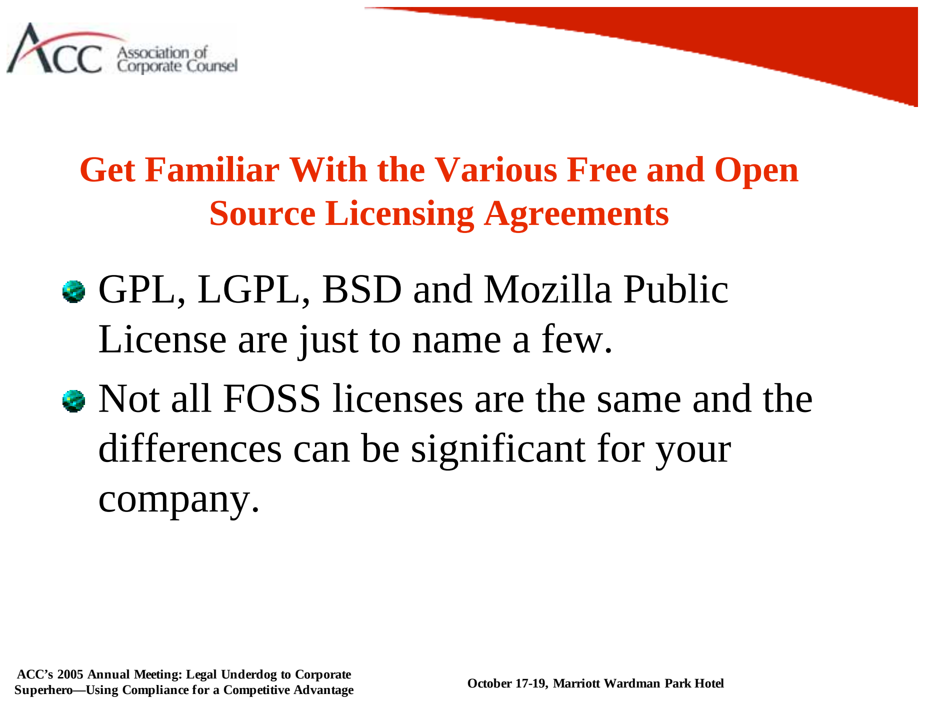
- Economical
  - Don’t have to pay for proprietary software or spend the time and resources in developing your own code.
- Efficient
  - Readily available and usually easy to implement.
  - Already has been tested by many other developers.
- Flexible
  - Most FOSS licenses allow you to modify the code to fit your business needs.





So Its Your Job to Limit Your Company's Risk

- Get familiar with the various FOSS licensing agreements.
- Be active in your company's software development from day one.
- Stay involved!

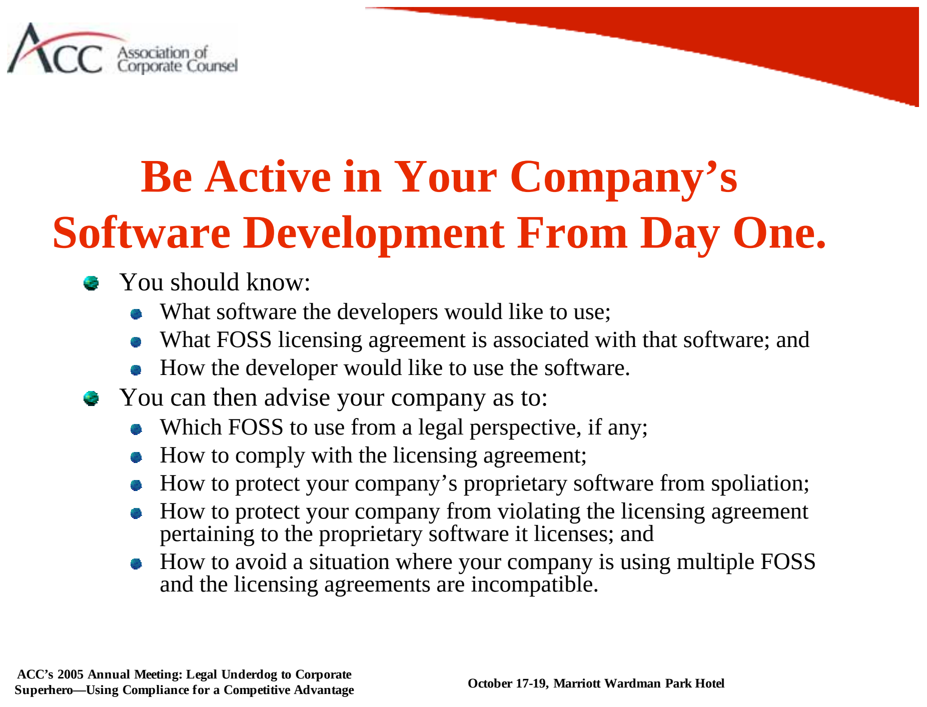


Get Familiar With the Various Free and Open Source Licensing Agreements

- GPL, LGPL, BSD and Mozilla Public License are just to name a few.
- Not all FOSS licenses are the same and the differences can be significant for your company.

ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage

October 17-19, Marriott Wardman Park Hotel

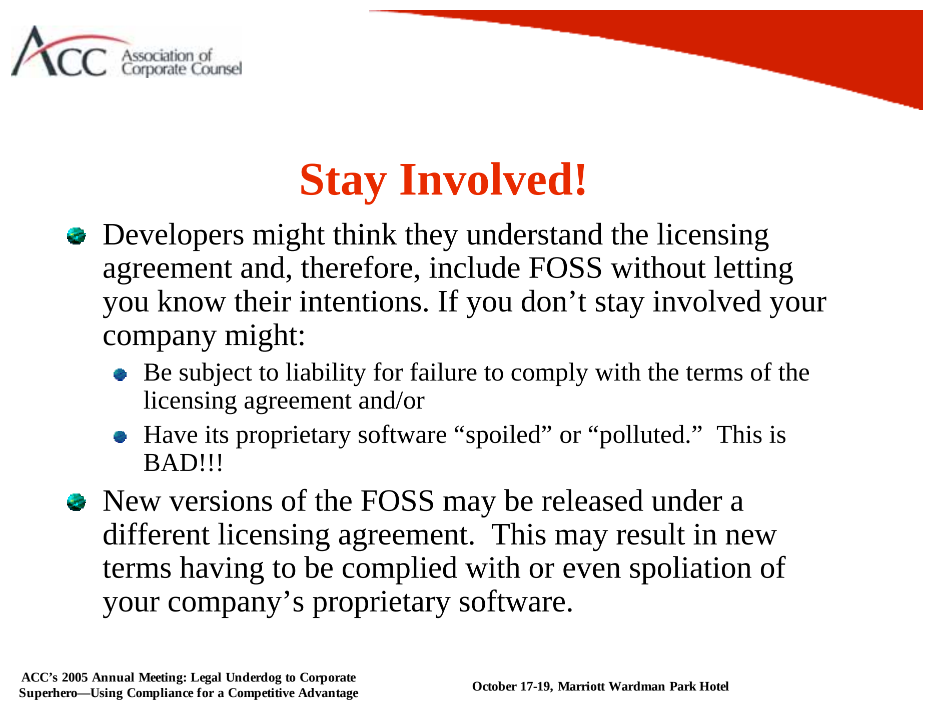


Be Active in Your Company's Software Development From Day One.

- You should know:
  - What software the developers would like to use;
  - What FOSS licensing agreement is associated with that software; and
  - How the developer would like to use the software.
- You can then advise your company as to:
  - Which FOSS to use from a legal perspective, if any;
  - How to comply with the licensing agreement;
  - How to protect your company's proprietary software from spoliation;
  - How to protect your company from violating the licensing agreement pertaining to the proprietary software it licenses; and
  - How to avoid a situation where your company is using multiple FOSS and the licensing agreements are incompatible.

ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage

October 17-19, Marriott Wardman Park Hotel



Stay Involved!

- Developers might think they understand the licensing agreement and, therefore, include FOSS without letting you know their intentions. If you don't stay involved your company might:
  - Be subject to liability for failure to comply with the terms of the licensing agreement and/or
  - Have its proprietary software "spoiled" or "polluted." This is BAD!!!
- New versions of the FOSS may be released under a different licensing agreement. This may result in new terms having to be complied with or even spoliation of your company's proprietary software.

ACC's 2005 Annual Meeting: Legal Underdog to Corporate Superhero—Using Compliance for a Competitive Advantage

October 17-19, Marriott Wardman Park Hotel

Open Source Software, Software Patents, and the Inevitable Collision

Mark H. Webbink
Deputy General Counsel, Red Hat, Inc.

Patent Law in the U.S. and the Evolution of Software Patents

Patents protect inventions, including improvements to those inventions. Such inventions may take the form of either a physical thing, e.g., a machine, the application of an idea, e.g., a process, or a design for an article of manufacture. Patents do not protect unapplied ideas, and they do not protect expression. In order to receive patent protection, an invention must be useful, novel (i.e., original and not the subject of prior art), and non-obvious.[1]

Patents grant the patent holder the right to exclude others from making, using, offering for sale, or selling" the invention in the United States or "importing" the invention into the United States.[2] Note, patents do not grant the holder the right to make, use, offer for sale, sell, or import, only the right to exclude others from doing so. This is so because any one patent holder may be subject to the patent rights of another patent holder. The classic example is of the patent holder for bicycle bells. While the holder of the bell patent may exclude others from making, using or selling bicycle bells, that holder is subject to the rights of the holders of patents in bells and bicycles, if any.

The national intellectual property office of each country or sovereign region issues patents. A patent granted in one country or sovereign region has no effect in another country or sovereign region, apart from possible priority in extending patent protection to such other country or sovereign region. As a result, a party seeking to protect patentable subject matter must apply for that protection in each country or sovereign region where they desire protection. A further result is that infringement can largely only be asserted in those countries or sovereign regions where such protection has been obtained. In addition, there is no assurance that inventions that are protected under U.S. patent law may be protected in such other jurisdictions. For example, computer software is not protected under the patent laws of most countries or sovereign regions.

[1] U.S. Patent and Trademark Office, *General Information Concerning Patents*, available at <http://www.uspto.gov/web/offices/pac/doc/general/index.html#patent> (last visted July 5, 2003).
[2] *Id.*

While U.S. patent law requires a patent application to sufficiently disclose the invention in its best mode to enable a person skilled in the applicable art the ability to repeat the invention, patent applications for software patents do not require the disclosure of any source code that implements the invention. This lack of disclosure can hamper the ability of competitors to identify exactly what the patent holder is claiming. Further, U.S. patent law does not require the publication of patent applications for at least 18 months following filing, thus denying competitors the ability to identify pending claims. Patent protection in the U.S. generally extends for a period of 20 years from the date of filing.

Patent protection has not always been available for computer software. In fact, prior to a number of court cases in the early 1980's it was well accepted that one could not obtain a patent on software. That wall began to crack with the *Diamond v. Diehr*[3] case in 1981 where the U.S. Supreme Court held that: ". . . a claim drawn to subject matter otherwise statutory does not become nonstatutory simply because it uses a mathematical formula, computer program, or digital computer."

Concurrent with the *Diamond* case were a series of federal court cases from 1978 to 1982 that articulated a test to determine the patentability of mathematical algorithms.[4] These cases resulted in the so-called *Freeman-Walter-Abele* test: "It is first determined whether a mathematical algorithm is recited directly or indirectly in the claim. If so, it is next determined whether the claimed invention as a whole is no more than the algorithm itself; that is, whether the claim is directed to a mathematical algorithm that is not applied to or limited by *physical elements or process steps.*[5] In the subsequent *Alapat* case the court held that: ". . . data, transformed by a machine through a series of mathematical calculations to produce a smooth waveform display on a rasterizer monitor, constituted a practical application of an abstract idea (a mathematical algorithm, formula, or calculation), because it produced "a useful, concrete and tangible result"--the smooth waveform."[6] That is, a patent claim which utilizes a mathematical algorithm to produce a "useful, concrete and tangible result, is, in fact, patentable.

3 Diamond v. Diehr, 450 U.S. 175, 181 (1981).
4 *See* In re Freeman, 573 F.2d 1237 (C.C.P.A. 1978); In re Walter, 618 F.2d 758 (C.C.P.A. 1980); and In re Abele, 684 F.2d 902 (C.C.P.A. 1982).
5 Arrhythmia Research Technology, Inc. v. Corazonix Corp., 958 F.2d 1053, 1058 (Fed. Cir. 1992)(emphasis added).

The final barrier to matters of patenting software fell with the *State Street Bank & Trust Co. v. Signature Financial Group*[7] holding in 1998 that: ". . . the mere fact that a claimed invention involves inputting numbers, calculating numbers, outputting numbers, and storing numbers, in and of itself, would not render it non-statutory subject matter, unless, of course, its operation does not produce a "useful, concrete and tangible result. . . .The question of whether a claim encompasses statutory subject matter should not focus on which of the four categories of subject matter a claim is directed to--process, machine, manufacture, or composition of matter--but rather on the essential characteristics of the subject matter, in particular, its practical utility." Thus, in a period of less than 20 years the U.S. has moved from a patent system that denied protection to software to one that embraces such protection.

Software Patents and Innovation

One of the widely accepted truths about software patents is that they are necessary as an incentive for innovation. However, all evidence points to the contrary. Rather, software patents are a prime source of mischief within the software industry.

Bill Gates recognized that software patents may prove to be problematic when, in 1991, he stated in a memorandum:

> "If people had understood how patents would be granted when most of today's ideas were invented, and had taken out patents, the industry would be at a complete standstill today. . . . I feel certain that some large company will patent some obvious thing related to interface, object orientation, algorithm, application extension or other crucial technique."[8]

Had software patents been around since 1975, the following inventions would have been subject to patents for the better part of the last 20 years: (a) WordStar, the first PC-based word processor released in 1979; (b) VisiCalc, the first PC-based spreadsheet program released in 1979; and (c) Harvard Graphics,

6 In re Alappat, 33 F.3d 1526, 1540-41 (Fed. Cir. 1994).
7 State Street Bank & Trust Co. v. Signature Financial Group, 149 F.3rd, 1368, 1374-1375 (Fed. Cir. 1998).
8 Bill Gates, *Challenges and Strategy*, (May 16, 1991), *available at* http://discuss.sarahsbookstores.com/Bill_Gates_Challenges_And_Strategy_Memo.

the first PC-based presentation graphics program released in 1983. Where would we be today had we been locked into only those choices? Needless to say, Microsoft Office would not be ubiquitous. Interestingly, Dan Bricklin, one of the inventors of VisiCalc, regularly writes and speaks today about the problem with software patents.[9] In 1991, when Bill Gates made his remarks, Microsoft had fewer than 50 filed patent applications; today Microsoft has over 4,000 issued patents and more than 10,000 pending patent applications.[10]

To appreciate what this means, one must understand the extent of the problem. It is estimated that there are more than 150,000 patents[11] on various forms of software or, more specifically, software features. While it is typical for a pharmaceutical product to be covered by a single patent,[12] that is not typical of software where hundreds if not thousands of patents may cover a single application. For example, Microsoft has 14 separate patents on the positioning and movement of a cursor,[13] and they have two additional applications pending on the same.[14] Rather than producing broad innovations to advance the software industry, like the earlier-mentioned inventions such as the word processor, spreadsheet, or presentation graphics, information is being sliced and diced to the point that every trivial combination or extension of prior software technology is being accorded the same protection as a groundbreaking drug. In the summer of 2004, when Bill Gates announced that Microsoft would be increasing its annual patent filings from 2,000 to 3,000 per year, it was notable that there was no corresponding 50% increase in Microsoft spending for research and development. In fact, an examination of the history of Microsoft demonstrates that patents played no meaningful role in the company's development prior to 1995.[15] It should be evident that this system of protecting software is not the root cause of innovation in the software industry.

9 Dan Bricklin, *Patents and Software*, *at* http://www.bricklin.com/patentsandsoftware.htm (last viewed October 8, 2004).
10 As identified by searching Microsoft, as assignee, against the U.S. Patent and Trademark database utilizing the patent and patent application advanced search functions found at http://www.uspto.gov/patft/index.html.
11 Based on all patents issued in classes 345, 700-707 and 715-717, which cover the bulk of software patents, as identified by searching the Current U.S. Classification for these classes against the U.S. Patent and Trademark database utilizing the patent and patent application advanced search functions found at http://www.uspto.gov/patft/index.html.
12 Pfizer's blockbuster, multi-billion dollar Viagra is covered by just one patent, as identified by searching the term "Viagra" utilizing the search tool found at http://www.fda.gov/cder/ob/docs/querytn.htm. Similarly, just one patent covered Merck's blockbuster, multi-billion dollar Zocor, as identified by searching the term "Zocor" utilizing the search tool found at http://www.fda.gov/cder/ob/docs/querytn.htm.
13 As identified by searching Microsoft as assignee and the term "cursor" in the title of issued patents against the U.S. Patent and Trademark database utilizing the patent advanced search functions found at http://www.uspto.gov/patft/index.html.
14 As identified by searching Microsoft as assignee and the term "cursor" in the title of patent applications against the U.S. Patent and Trademark database utilizing the patent application advanced search functions found at http://www.uspto.gov/patft/index.html.

15 Mark Webbink, *A New Paradigm for Intellectual Property Rights in Software*, 2005 Duke Law & Technology Review 0012, *available at* http://www.law.duke.edu/journals/dltr/.

It is bad enough that software patents are being filed at an astounding rate for such trivial matters, but the lack of scrutiny such patents receive, is telling as well. Software patents are not the only class of patents that are vulnerable to reexamination and invalidation; patents of all classes are highly vulnerable to such assertions. In their report entitled *Empirical Evidence on the Validity of Litigated Patents*, John R. Allison and Mark A. Lemley of the University of Texas examine patent validity opinions from an 8-year period through 1996.[16] They found that fully 46% of all patents litigated were invalidated.[17] When only software patents were considered, two-thirds of them were invalidated.[18] This comes as no surprise to anyone who has spent time reviewing software patents. The lack of an established and easily accessible body of prior art, relaxed application of standards of non-obviousness, and pressure on the USPTO examiners to meet prosecution performance statistics have all contributed to this condition.

The problems presented by software patents and their negative impact on innovation have not gone unnoticed. As pointed out in their paper entitled *The Software Patent Experiment*, James Bessen and Robert Hunt argue that there is reasonable evidence to show that software patents are not inducing innovation.[19] They found that established firms obtain most software patents and to a greater degree than established firms in other industries. Interestingly, they also found a negative correlation between increases in a firms' software patent focus and their R&D intensity, suggesting that such established firms are substituting software patents for R&D. This is further born out by Bessen and Hunt in their finding that:

> "[Where companies are assembling large portfolios of software patents, such] firms may compete to tax each others' inventions and in the process reduce their competitors' incentive to engage in R&D. The outcome of patent litigation and licensing agreements often depends on the size of the firm's patent portfolio. This creates an incentive to build larger patent portfolios, especially when the firm focuses on patents as a competitive strategy. In this account, firms choose to compete in court, rather than in the marketplace."[20]

They go on to hypothesize:

16 John R. Allison & Mark A. Lemley, *Empirical Evidence on the Validity of Litigated Patents*, 26 AIPLA Q.J. 185, 205-206 (1998), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=118149.§
17 *Id.* at 16.
18 *Id.* at 17.
19 James Bessen & Robert M. Hunt, *The Software Patent Experiment*, 2 (March 16, 2004), *available at* http://www.researchoninnovation.org/softpat.pdf.
20 *Id.* at 14-15.

> "During the early 1980s, patents were relatively costly to obtain, and this might have discouraged substitution away from R&D and toward strategic patenting. By the mid 1990s, software patents became a relatively inexpensive way to expand patent portfolios. This may have increased the attractiveness of a strategy that emphasizes patent rights over a strategy based on R&D. Such a change in strategy would be particularly attractive to mature firms if their R&D labs are not as productive as they once were."[21]

In a 2004 research report prepared by Deutsche Bank Research, the authors discuss the issue of Germany's lag in introducing technology innovation.[22] While the report calls for properly valuing intellectual property as one step in increasing innovation, it also calls for "a balanced IP protection regime to foster the creation and flow of ideas," going on to state that "stronger IP protection is not always better. Chances are that patents on software, common practice in the US and on the brink of being legalised in Europe, in fact stifle innovation. Europe could still alter course."[23] Citing the findings and suggestions of a study by James Bessen and Eric Maskin,[24] Deutsche Bank recommends favoring copyright protection over patent protection for software as a means of maintaining a more level playing field and attracting and inviting innovation from the sector that historically has produced it—the small- and medium-sized enterprise. It is significant that Europeans are recognizing the flaws in the U.S. system.

This sentiment is further echoed in the August 2004 report by PricewaterhouseCoopers for the Dutch Ministry of Economic Affairs entitled *Rethinking the European ICT Agenda*.[25] In that report, PricewaterhouseCoopers states:

> "There are particular threats to the European ICT [Information and Communication Technology] industry such as the current discussion on the patent on software. The mild regime of IP protection in the past has led to a very innovative and competitive software industry with low entry barriers. A software patent, which serves to protect inventions of a non-technical nature, could kill the high innovation rate."[26]

21 *Id*. at 15.
22 Deutsche Bank Research, *Current Issues – More Growth for Germany*, (June 22, 2004), *available at* http://www.dbresearch.com.
23 *Id*.
24 "Sequential Innovation, Patents, and Imitation" - James Bessen and Eric Maskin – November, 1999 (www.researchoninnovation.org/patrev.pdf)
25 PricewaterhouseCoopers, *Rethinking the European ICT Agenda*, (August 2004) *available at* https://www.ictstrategy-eu2004.nl/pdf/Rethinking_the_European_ICT_agenda_def.pdf.
26 *Id*. at 50.

In another empirical study reported by Petra Moser of MIT in the 2003 paper entitled *How Do Patent Laws Influence Innovation? Evidence From Nineteenth-Century World Fairs*,[27] Moser finds no evidence that patent laws increased levels of innovative activity. Rather, she reveals strong evidence that patent systems influenced the distribution of innovative activity across industries. In fact, evidence presented by Moser substantiates the contention that countries without patent laws were just as innovative as those with strong patent protection. Moser's findings are further supported in Bronwyn Hall's 2003 paper, *Business Method Patents, Innovation and Policy*.[28] Hall reaches two conclusions: "(1) there exists a unique standard of nonobviousness that maximizes the rate of innovation in a given industry; and (2) contrary to the conventional wisdom, reductions in the nonobviousness requirement are more likely to encourage innovation in industries that innovate slowly than in industries that innovate rapidly."[29] She goes on to state: "The implication is that in rapidly innovating industries where each new product builds on others, welfare is more likely to be enhanced by having a high hurdle for obtaining a patent."[30]

Proprietary vs. Open Source Licenses – Can Patents Tell the Difference?

Do patents distinguish between software licensed under a proprietary license and software licensed under an open source license? This is actually a fairly silly question. Of course they do not. Software patents are just as problematic to proprietary software vendors as they are to developers of open source software. As the largest player in the software industry, Microsoft typically faces 30 or more patent infringement actions at any one time and spends $100 million a year defending itself in patent infringement litigation.[31] This is not because Microsoft does not try to avoid infringing the patents of others; they are simply a highly lucrative target. By the same token, the fact that open source software has faced relatively few patent infringement claims to date does not mean that open source software is immune to such claims. Everyone in the software industry must be cognizant of the threat that software patents pose.

27 Petra Moser, *How Do Patent Laws Influence Innovation? Evidence From Nineteenth-Century World Fairs*, (August 2003), *at* https://www.nber.org/papers/w9909.
28 Bronwyn Hall, *Business Method Patents, Innovation, and Policy*, (May 4, 2003), *at* http://repositories.cdlib.org/iber/econ/E03-331/.
29 *Id*. at 7.
30 *Id*. at 8.
31 Lisa DeCarlo, *Microsoft to the Rescue?*, Forbes, march 10, 2005, http://www.forbes.com/2005/03/10/cx_ld_0310msft_print.html.

Open Source Licenses and Patents

Open source licenses, which are for the most part copyright licenses, essentially address patents in one of three ways: directly, obliquely and not at all. Open source licenses that do not address patents at all are the most common as exemplified by the Berkeley Software Development (or BSD) license.[32] The BSD simply makes no reference to patents. Consequently, whether the BSD or similarly licenses convey any inferential rights in patents of the licensing party is subject to question. One could certainly argue that a BSD licensor, by offering broad rights to use, modify and redistribute the BSD-licensed code, has granted an implied license in any patents such BSD licensor holds that read on the BSD-licensed code as distributed. That uncertainty may not be acceptable to some BSD licensors, and Intel, for one, chose not to leave this question to chance when it introduced its BSD + Patent license.[33]

Other open source license directly address patent rights, including the termination of those rights. Examples are the Mozilla Public License,[34] the Common Public License,[35] the IBM Public License[36] and the Open Software License.[37] These licenses typically include a grant of a patent license from each code contributor to the patents of such contributor that are necessarily infringed by the contributor's code contribution to the project. A couple of things to note about these license grants. First, they are not as expansive as the copyright license grants included in the same license because they do not necessarily grant rights to new patent claims that may be infringed by the contributions of other parties. Second, they typically include a termination of such patent rights to a licensee if such licensee brings a patent infringement suit against the licensor.

The GNU General Public License (the GPL),[38] the most widely used of all open source licenses, treats patents in a far more oblique manner. Section 7 of the GPL states:

[32] http://www.opensource.org/licenses/bsd-license.php
[33] http://infiniband.sourceforge.net/duallicense.htm
[34] http://www.opensource.org/licenses/mozilla1.1.php
[35] http://www.opensource.org/licenses/cpl1.0.php
[36] http://www.opensource.org/licenses/ibmpl.php
[37] http://www.opensource.org/licenses/osl-2.1.php
[38] http://www.opensource.org/licenses/gpl-license.php

> "If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
>
> If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
>
> It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
>
> This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License."

Although the Free Software Foundation, the promoter of the GPL, offers a robust FAQ site that provides interpretation of many of the sections of the GPL,[39] the FSF does not provide an interpretation of exactly what Section 7 means. There are at least three situations that could use such interpretation: (a) where the GPL-licensor is also the patent holder; (b) where the GPL-

[39] http://www.fsf.org/licensing/licenses/gpl-faq.html

licensor holds no patents but is aware of third-party patents that read on the licensed code; and (c) where third-party patents read on the licensed code and the GPL-licensor holds a patent license from that third-party. Let us examine each of these in relation to Section 7.

Where the GPL licensor holds patents that read on the GPL-licensed code, the following language is instructive:

> "If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program."

This provision has generally been interpreted to mean that, if you hold patents that read on code that you are licensing under the GPL and you distribute that GPL-licensed code to others, you have granted all those recipients, whether direct or indirect, an implied license in your patents to the extent they read on the code as you distributed it. Whether that implied license extends to modifications to the code made by others is unclear, but given that those others, not you, are the distributors of such modifications, it is not illogical to assume that they, not you, are the party who must independently comply with the requirements of Section 7 with respect to those modifications.

The second case, where unlicensed patents held by a third-party of which the GPL licensor is aware and that read on the GPL-licensed code, is a bit more difficult. One can argue that the mere existence of a third-party patent that has not been asserted does not constitute the imposition of a condition that contradicts the terms of the GPL. It is merely a latent condition that has the potential of arising at such future time but that does not constitute a condition at present. An unasserted third-party patent certainly does not constitute a present obligation on the GPL licensor.

In the third case, where third-party patents read on the GPL-licensed code and are licensed to the GPL licensor, compliance is largely determined by the terms of the agreement between the patent licensor and the GPL licensor. If that agreement is a patent license to the GPL licensor with no right to sublicense others, it would appear to contradict the terms of Section 7. On the other hand, should the agreement be structured as a covenant not to sue, rather than a license, and the GPL licensor has the right to extend that covenant to the GPL licensor's licensees, perhaps in the form of indemnification for which the licensee may be required to compensate the GPL licensor,[40] then the GPL licensor may still be in compliance with Section 7. Why? Because the GPL licensor has shielded the indemnified licensee from patent infringement claims, and the GPL makes clear that any GPL licensor has no obligation to extend warranty or, by implication, indemnification protection to any GPL licensee. In any case, a GPL licensor needs to be aware of and address these and other possible scenarios surrounding software patents.

<u>Conclusion</u>

Software patents have little to do with innovation. They were not critical to the development of the software industry, and they are primarily being used today by large industry players to maintain market position. Arguably, software patents slow the rate of innovation rather than accelerating it.

Software patents are difficult with which to deal. Many tend to cover relatively minor inventions or incremental extensions of those inventions. The shear number of software patents and the rate by which they are expanding make it almost impossible for a software developer to know with assurance that someone else's patent does not read on his/her new code. The cost of litigating patents is high and adds to this burden. Further, software patents are ignorant of software licensing models and are a burden on both proprietary and open source software licensors.

[40] *See*, GPL Section 1 which 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."

Finally, open source licenses address patents in a variety of ways. It behooves any open source licensor and licensee to be aware of whether patent rights are granted, and the extent of such rights.