White Paper:

Open Source Software Licenses: Perspectives of the End User and the Software Developer

By: Paul H. Arne
Morris, Manning & Martin, L.L.P.

MORRIS, MANNING & MARTIN, LLP
ATTORNEYS AT LAW

Table of Contents

Open Source Software Licenses: Perspectives of the End User and the Software Developer[1]

By: Paul H. Arne[2,3]
Morris, Manning & Martin, L.L.P.

Fear, uncertainty, and doubt are major factors when a company first considers using open source software. Programmers tend to love open source software. They say it's stable, cheap, and available online. They may already be contributors to the open source movement. It saves them time. It is much less expensive than other alternatives. Yet CIO's and business executives hear other stories that give them pause about using open source software. They hear that some companies are very publicly opposed to it. They hear of lawsuits and threatened lawsuits. Frequently, attorneys within companies are opposed to the use of open source software for reasons that are heartfelt and seemingly logical.

This white paper reduces the "FUD" factor by clarifying the legal issues associated with open source licensing. In most ways, open source licenses are just like any other license to software. With an understanding of how intellectual property laws work and how software is built, one reads the license agreement to determine what a licensee can and cannot do, what the licensee is required or not required to do, and how the risks have been allocated. There are some activities for which companies should not use some open source software. In other situations, many companies will decide that the use of open source software is perfectly acceptable.

This white paper is directed to two audiences: (i) those who want to use open source software with little or no modification and only for internal use ("End Users") and (ii) those software developers who want to use open source software as a part of a software package that will be sold[4] to others ("Developers"; Developers are frequently referred to as Independent Software Vendors (or ISVs), system integrators, value added resellers, original equipment manufacturers, and independent contractors, among other names). Some of the issues discussed in this white paper are legally technical (after all it *is* a white paper), but the risks identified or clarified are not technical — they're just business risks. If you get bogged down in the text, just skip a few sentences.

---

[1] This document does not create an attorney/client relationship with you and does not provide specific legal advice to you or your company. Certain legal concepts have not been fully developed and certain legal issues have been stated as fact for which arguments can be made to the contrary, due to space constraints. It is provided for educational purposes only.

[2] Paul Arne is a partner with Morris, Manning & Martin, and chairs its Open Source practice group.

[3] Special thanks to Lauren Sullins, who assisted with the research and writing of this white paper.

[4] Technically, the software is licensed, not sold.

History of Open Source[5]

The open source movement got its start in the early 80's with the development of a "freeware" version of UNIX, known as GNU, developed by Richard Stallman. Stallman was a strong advocate that software should be free.[6] From that development came the first open source license, the GNU General Public License, or "GPL."

There is some brilliance to the design of the GPL. One might think that a free software movement would seek to avoid copyright laws, the principal intellectual property law that restricts the free use of software and allows its commercial exploitation. Instead, the GPL uses the rights under copyright to enforce the goals of the free software movement. Not surprisingly, Stallman refers to his licensing scheme as "copyleft." It really does turn normal copyright on its head.

Probably the most important development in the open source movement has been the development of Linux, a popular open source operating system. Linus Torvalds, the developer of the Linux kernel, used the GPL as the licensing scheme for Linux. In part because the GPL, unlike many other open source licenses, requires modifications that are distributed to be shared, source code and all, literally thousands[7] of programmers have been involved in writing, editing, and testing Linux. Linux has developed a well-deserved reputation as being a very stable, inexpensive operating system platform that can run desktops to supercomputers. In addition, the Linux community typically responds quickly to changes in the landscape of computing, adding capabilities or drivers rapidly after the introduction of the corresponding hardware.

Businesses have grown to provide services around Linux. In addition, a significant number of other individuals and companies have adopted the open source model to develop their products.[8]

It is important to note that open source is not a single movement or a single form of software license. The Open Source Initiative[9] lists approximately 50 different "open source" licenses. These licenses vary significantly in their requirements. Therefore, it is critical to obtain and actually read the license that governs the use, modification, and distribution of the software in question.

---

[5] For more detailed information about the origins and history of the open source movement, see www.fsf.org/gnu/thegnuproject.html (the history of GNU), https://netfiles.uiuc.edu/rhasan/linux/ (the history of Linux), and www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ (a sweeping analysis of the open source movement).

[6] One of Mr. Stallman's writings on this topic can be found at www.fsf.org/philosophy/shouldbefree.html.

[7] "Published estimates range from several hundred to more than 40,000." Essence of Distributed Work, the Case of the Linux Kernel, by Jae Yun Moon and Lee Sproull, www.firstmonday.org/issues/issue5_11/moon/index.html.

[8] As of this writing, one Web site currently hosts over 83,000 open source software projects. See www.sourceforge.net.

[9] A non-profit organization that exists to promote open source. Among other things, OSI "certifies" software licenses as being official open source licenses. Its Web site is at www.opensource.org.

Open Source Licenses Generally

Some open source license agreements seem to have been drafted by engineers as well as lawyers. These licenses can display a technical sophistication that usually does not exist in run-of-the mill license agreements. Even for lawyers who practice regularly in this field, understanding an open source license can be challenging. In addition, some of the licenses do not follow conventions of tight legal drafting, giving rise to some unnecessary uncertainty in the meaning of the licenses. Finally, because copyright law generally and the concept of derivative works specifically are often used in open source license agreements to dictate whether certain rights or obligations exist, a good understanding of copyright law and what a derivative work is and is not can be important.

Copyright Issues

Copyright law gives specific rights to holders of copyrights. These "exclusive" rights may be granted by license or sale to third parties at the discretion of the copyright holder. The exclusive rights include the following:

- to make copies of the work,
- to prepare derivative works of the work, and
- to distribute the work.[10]

This legal structure is critically important to understand. It is the basis for the sale of books, music, movies, and software. As long as the rights are appropriately reserved by the copyright holder, the sale or license of a book, a CD, a DVD, or software does not grant to the recipient any of the exclusive rights under copyright law, specifically the right to make copies, to create derivative works, and to distribute[11] the work.

These concepts serve as the basis for some open source licenses. The limitations stated in the open source license agreement are enforceable against a user, even when the user doesn't agree to the terms or manifest any intention to be bound, because the rights that are enforced are copyrights, not contract rights. Because of the exclusive rights granted to the copyright holder, a copyright holder can limit uses of the software by simply not giving those rights to the recipient. A copyright holder can also place conditions on the right to do certain things. Any use of the software beyond the limits or without complying with the conditions set forth in the license exceeds the scope of the licenses granted and is therefore a copyright infringement. In essence, limitations

[10] 17 U.S.C. § 106 (2004).

[11] Note that the "first sale doctrine" allows the subsequent distribution of a copy of a work that is sold, not licensed. This means that you can sell or give a copy of a book, CD or DVD to someone else, so long as you don't make copies. Preventing subsequent sale is the reason that software is licensed and not sold. Examination of the first sale doctrine is beyond the scope of this white paper, as are other rights under copyright that may relate to the rights of owners of copyrights and owners of copies of copyrighted works, such as public performance rights.

of a license can be enforced without the necessity of an actual agreement by simply not licensing those rights to the recipient or otherwise placing restrictions on certain rights. This is exactly what happens with a book. You don't need a contract with the author for the author to prohibit your copying of the book; the author simply doesn't give you that right.

Generally, an open source license will place requirements on the user in order for the user to have the right to copy, distribute, or create derivative works of the software. There are a number of types of restrictions that can be found in open source licenses. They include:

- Requirements regarding the placement of copyright notices on copies;
- Requirements that notices be given of modifications to the software;
- Requirements that the software either be or not be attributed to certain authors;
- Requirements that certain disclaimers of liability be included in any subsequent license;
- Requirements that certain limitations of damages be included in any subsequent license;
- Requirements that the software and all modifications be distributed only with the source code or an offer to provide the source code for free, other than a copying fee; and
- Requirements that the object code be distributed for free or only for a reasonable copying charge.

The last two requirements above, the distribution of modified source and object code for only a copy fee, are normally what give companies the most concern. This issue will be explored in more detail in the discussion of the GPL below. For the reasons set forth in the GPL discussion, suffice it to say that these requirements are likely to be of little concern to an End User, especially if the open source software isn't modified by the End User.

Contract Considerations

Most open source licenses contain other provisions that are harder to characterize as some exclusive right that the copyright holder is retaining rather than giving away. Examples of these include limitations of liability and disclaimers of implied warranties, specifically the warranties of merchantability and fitness for purpose implied in some contracts under the Uniform Commercial Code. It is harder to argue that a limitation of liability is a limitation on the ability to make copies, create derivative works, or distribute the software. Accordingly, in order to be enforceable, other laws

must probably be relied on other than copyright in order to render them enforceable.[12] Contract law is the likely alternative.

It is not clear from the way open source software and their corresponding licenses are frequently distributed that a binding contract actually exists.[13] Having an enforceable contract normally requires parties to actually agree to something;[14] a contract cannot be unilaterally imposed by one party on another. Typically, when open source software is downloaded off of the Internet, the Web page lists the open source license by name, without a link to its terms, without the text of the license being displayed, and without requiring the downloading party to assent to the license at the time of the download. Accordingly, there may be no manifestation of agreement or assent on the part of the user that would be required to create a legally enforceable contract.[15] This suggests that provisions of open source licenses that aren't self-executing in their operation may not be enforceable.[16]

For an End User, this is not much of an issue. An End User would view the unenforceability of a limitation of liability clause or disclaimer of warranties as a good thing.

Developers are similarly helped rather than hindered. However, Developers should consider this issue when they distribute the open source to a customer. The simple solution for a Developer is to create an enforceable contract when it delivers the open source, along with other Developer software. By incorporating certain provisions of the open source license into an enforceable license, the Developer should be able to avoid potential problems with the lack of enforceability of parts of the open source license.

Limitation of Liability Clause

The disclaimers of liability in open source licenses typically disclaim *all* damages, not just the more standard disclaimers of consequential and special damages. One can argue that an otherwise valid contract is rendered invalid when one party has no

---

[12] Where the scope of copyright law ends in this context is not clear. There may be situations where the nature of the restriction may not be clearly a copyright-based limitation or a limitation requiring a contract to be enforceable.

[13] In certain seminars, representatives of the Free Software Foundation have indicated that the Free Software Foundation Licenses (most notably the GPL and LGPL) are not contracts but are only licenses.

[14] Restatement of Contracts, 2nd, §19. The point of this discussion is not to show without a doubt that a contract does not exist; it is only to show that there is a potential argument that contract rights do not exist under the circumstances of a typical download and use of open source software.

[15] See, e.g., Specht v. Netscape Communs. Corp., 306 F.3d 17 (2d Cir. 2002), holding that there was no mutual assent when the terms of the license governing the use of the software to be downloaded appeared "below the fold" (i.e., requiring the user to scroll down the Web page).

[16] This is easy to fix, however. As a part of the distribution process, making the receipt of a copy conditioned on clicking an "I accept" button while having access to the terms of the license agreement would likely do the trick. While there are a few additional requirements, in most countries it is not hard to create an enforceable agreement entered into exclusively online.

right to damages for breaches of the agreement by the other party.[17] This lack of "mutuality" is another potential issue with most of the open source licenses. This problem is the licensor's problem, however, not the End User's. It is nevertheless something to consider when a Developer distributes open source software with its own software.

Other Implied Warranties

There may be more implied warranties than merchantability and fitness for purpose. There are plausible arguments that licenses of some software also come with implied warranties of title and noninfringement. Not all open source licenses disclaim these warranties. However, note that the absence of a valid disclaimer may help, and certainly shouldn't hurt, the End User and Developer. Developers should consider this issue when licensing the open source software downstream.

UCITA

Virginia and Maryland have adopted the Uniform Computer Information Transactions Act ("UCITA"). In UCITA, there are additional implied warranties that are factored into software licenses. These warranties include the warranties of noninfringement, noninterference, and system integration, as well as certain data-related warranties. Generally, open source licenses do not disclaim these warranties. Accordingly, they may be available to assert in connection with problems related to the software where the governing law is Virginia or Maryland.[18] Once again, the absence of disclaimers of these warranties doesn't hurt the End User and Developer; if anything, it helps.

Parties to License

Because most open source software is developed by multiple people and companies, determining who the licensor actually is can be difficult, if not impossible, to determine. The open source licenses reviewed in this white paper do not contain assignments of contributor's intellectual property rights to a single organization, so individual contributors may retain intellectual property rights to various parts.

The existence of multiple contributors and the lack of a central licensor has at least a couple of ramifications. First, practically there may be no person or organization that can sue an End User or Developer who violates the terms of an open source license agreement.[19] Second, there may be no one for an End User or Developer to sue if that becomes necessary or appropriate.

---

[17] See Sterling Computer Systems of Texas, Inc. v. Texas Pipe Bending Co., 507 S.W.2d 282 (Tex. Civ. App. -- Houston [14th Dist.] 1974, writ ref'd); Spellman v. Lyons Petroleum, 709 S.W.2d. 295 (Tex. Civ. App. –Houston [14th Dist.] 1986)

[18] Virginia recently adopted amendments to UCITA that eliminated some warranties for "free software."

[19] Note that it is possible for a contributor to assign his or her intellectual property rights to the Free Software Foundation. The Free Software foundation has also created software on its own, such as GNU.

Specific Open Source Licenses

GNU General Public License (GPL)

General Conclusions

The GPL is the most important open source license to understand, simply because much open source software is licensed under it. Linux is licensed under the GPL. As of this writing, the SourceForge Web site[20] has over 38,000 software packages available for download that are licensed under the GPL.

Generally speaking, under the GPL any modifications of the code that you create and distribute are required to be distributed to all others in source code and object code form for only a fee for copying. Accordingly, and subject to the qualifications discussed below, software licensed under the GPL can be great for internal use only. Under normal circumstances, software licensed under the GPL should not, and probably cannot legally, be used for development of proprietary software for license to third parties for a license fee.

The GPL Generally

One of the most important features of the GPL is reflected in the following license provisions: "Activities other than copying, distribution and modification are not covered by this License [the GPL]; they are outside its scope." Despite being "outside the scope," the next sentence contains what seems to be a grant of the right to use the software (known as "Program" in the GPL). "The act of running the Program is not restricted…." This language suggests that the GPL gives the licensee the right to run the program in any way desired. There are no limitations regarding the number of users, log-in IDs, seats, named users, number of computers, kinds of computers, amount of data processed, etc.

In its general provisions, the GPL mentions restrictions on copying, distribution, and modification. When specific requirements are mentioned, however, they almost always address what happens upon distribution or modification, or both, and not what happens upon copying. There are virtually no provisions associated with copying the software outside of copying in connection with distribution. The emphasis of the GPL is clearly on the requirements related to distribution of modified copies of the software.

Section 1 of the GPL governs distributions of unmodified copies of the software. While there are requirements regarding copyright notices, disclaimers, providing copies of the GPL with the software and the like that must be complied with, there are no restrictions that would seem to concern either an

---

The Free Software Foundation has apparently pursued some violators of the GNU, but these have not gone to court. See www.fsf.org/philosophy/enforcing-gpl.html.

[20] www.sourceforge.net.

End User or Developer, as long as those entities do not want to make money on selling the software subject to the GPL.

Modifications and Distribution

Section 2 of the GPL addresses modifications to the software. The most important part of Section 2 provides, "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." Much of the rest of Section 2 addresses what this quoted language means.

For an End User who merely uses the software without modification, logically there can be no problem with this requirement. Indeed, even if an End User makes modifications to the software, Section 2 only comes into effect when there is a distribution or publication of the software. Internal use of modified software would not seem to require the free licensing of the software to third parties. Stated another way, there are no provisions in the GPL that *require* distribution of modified software. Only when modified software is distributed do requirements regarding free distribution and access to source come into play.

There are two caveats to the above, however. Case law in the software area makes clear that independent contractors of a licensee are not the licensee for licensing purposes.[21] Consequently, if your company obtains some of its "employees" from an employment service that remains the actual employer, engages consultants to work on projects, or otherwise uses 1099-type labor to provide IT services, then the use of the software by those persons technically might be considered a "distribution" under Section 2. If it is a "distribution," then under the GPL you must provide the software to them, in source and object code, for distribution by them under the terms of the GPL. Therefore, even if you are an End User, if you modify the software there may be a risk that you will be legally required to allow your modification to become a part of the open source community, freely useable by anyone for the asking.[22]

---

[21] MAI Systems Corp. v. Peak Computer, Inc., 991 F.2d 511 (9th Cir. 1993).

[22] This is actually a complicated issue. Section 2 of the GPL provides:

> 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:
>
> 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. (emphasis supplied)

Because of MAI v. Peak, it is fairly clear that a distribution has occurred from a legal standpoint.

However, the FAQ to the GPL, also prepared by the Free Software Foundation, has the following to offer:

> Q: Does the GPL allow me to develop a modified version under a nondisclosure agreement?

The second caveat is a variation of the first. Subsidiaries aren't the licensee for licensing purposes, either. As a result, providing modified code to a subsidiary, or other company within a controlled family of companies, may be considered a distribution subject to the provisions of the GPL.[23]

These caveats point out issues that exist with most software licenses, not just open source licenses. In the author's practice, it is a rare situation where a licensor's form software license agreement allows 1099 labor to use the software on behalf of a company that licenses the software. It is also a rare situation where an IT department of any size doesn't use 1099 labor. While more standard licenses allow use of the software by related companies, it is still unusual for a standard license prepared by a licensor to include this right.

Sophisticated end users seek to add these rights to software licenses, principally because they are risks and because they allow the license agreement to reflect what actually occurs in real life. Normally, licensors don't care about these technical issues, so they aren't likely to sue over a technical exceeding of the scope of the licenses even if these technicalities aren't fixed in the license agreement. It is not clear whether someone in the open source community would use these issues as a basis to insist that the modifications otherwise used exclusively internally by a company be disclosed to the public.

> A: Yes. For instance, you can accept a contract to develop changes and agree not to release your changes until the client says ok. This is permitted because in this case no GPL-covered code is being distributed under an NDA.
>
> You can also release your changes to the client under the GPL, but agree not to release them to anyone else unless the client says ok. In this case, too, no GPL-covered code is being distributed under an NDA, or under any additional restrictions.
>
> The GPL would give the client the right to redistribute your version. In this scenario, the client will probably choose not to exercise that right, but does have the right.

The FAQ also provides:

> Q: If I know someone has a copy of a GPL-covered program, can I demand he give me a copy?
>
> A: No. The GPL gives him permission to make and redistribute copies of the program if he chooses to do so. He also has the right not to redistribute the program, if that is what he chooses.

From the foregoing, apparently the obligation to distribute source code without restriction applies only to recipients (and therefore licensees) of the program subject to the GPL. Therefore, an independent contractor could develop code using the GPL under nondisclosure and provide it to his or her company for use and the company would not be obligated to distribute it.

However, the reverse seems not to be true. If the company shares the code with the independent contractor, then the company would not be able to limit the independent contractor's use of the code, even with a nondisclosure agreement. This makes co-development situations problematical under the GPL.

[23] This caveat is of lesser importance because a subsidiary probably won't feel compelled to distribute the modified open source software.

What's a Modification?

To understand the implications of Section 2 of the GPL, an understanding of the concept of derivative works is required. As stated above, the right to create derivative works is one of the exclusive rights held by the owner of a copyright. Whether a work is a derivative work of another is usually a question of whether you started by copying another's work. If your starting point is someone else's copyrighted work, either part of it or all of it, then almost without exception you are creating a derivative work.[24]

How Modifications are Handled

Section 2 states that if you modify the software (and then distribute it), then you are subject to the requirements of Section 2. These requirements include the following:

- Providing a notice that the software has been changed and when;
- Distributing the modified software "as a whole at no charge to all third parties" under the terms of the GPL; and
- Taking other actions in the event that certain operations automatically occur when the software is loaded, such as providing appropriate copyright notices on splash screens.

The GPL then takes three paragraphs to explain what is meant. The explanation is quoted below, with the author's explanation in brackets.

If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves [in other words, if you separate portions of the work and those portions are not derivative works of the Program], 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 [i.e., when the work as a whole remains a derivative work] 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 [i.e., when your work isn't a derivative work of the

[24] There are exceptions for short phrases, things that can only be expressed in a limited number of ways, and other exceptions, but none of those exceptions really take away from the basic principle stated above. See Pelt v. CBS, Inc., 1993 U.S. Dist. LEXIS 20464 (C.D. Cal. Oct. 29, 1993).

open source software]; rather, the intent is to exercise the right to control the distribution of derivative or collective[25] works based on the Program.

The GPL goes on to specify that distributing open source software with your software on the same media, such as a volume on a hard drive or optical disk, doesn't make your software a derivative work or subject to the GPL.

Section 3 of the GPL provides requirements related to the obligation to provide or offer to provide the source code of the software that is subject to the GPL along with any distribution of the object code.

Handling of API's and Other Interfaces

Software programs talk to each other. For example, in the event that an application program wants to save a file to a hard disk, the application program sends a command to the operating system software that in turn performs that operation. In order for the operating system to save the file to disk, however, the operating system must receive a command from the application software that it can understand. Since it must be understood by the operating system, the form of command must be dictated by the operating system. Frequently, this calling of functionality from one program to another is mediated through information known as an application program interface, or API. API's are formalized instructions for causing one program to use the functionality of another program. There are less formal ways for one software program to cause another software program to act, but the basic concept is that something in one program has to invoke something that is already in the other program.

If software uses the API or other interface of open source software, does the software become a derivative work of the open source software? This is one of the unanswered questions of the GPL; there are no express provisions of the GPL that address interfaces. Because the GPL does not answer this question directly, we are left with the question whether copyright law provides the answer. Under copyright law, there are no definitive answers, either. However, there are cases suggesting that where interoperability is concerned, courts will look upon

[25] The use of the phrase "collective works" is a troubling one. Collective works are different from derivative works — both are terms used in the Copyright Act. An example of a "collective work" is a book of short stories by different authors. Each author has a copyright to his or her individual contribution, while the publisher has the copyright in the collective group of short stories, subject to the rights of each individual author to authorize the use of his or her particular short story in the book. This suggests that under the GPL if you have a "collective work" of software, consisting of software subject to the GPL and software that is not subject to the GPL, when distributing the two software programs together both software programs are subject to the GPL. However, the GPL then states that distributing the software programs on the same media, comparable to distribution of two short stories in a book, does not require licensing the non-GPL software under the GPL. The apparent attempt of the GPL to require collective works to be distributed under the GPL — at least under some undefined circumstances but not others — is one of the most uncertain aspects of the GPL. The references to collective works in the GPL is a risk that Developers should consider when they distribute their proprietary software with bundled software subject to the GPL.

those situations with a view toward making interoperability available without violating another's exclusive rights under copyright.[26]

Based on these cases, the use of an API or other interface to open source software by non-open source software is not likely to result in the non-open source software being considered a derivative work, and therefore subject to the GPL. However, the answer is not certain. To some extent the answer may depend on how the software programs actually work together and how much of the open source software is actually incorporated into the other software program. To an End User, a company that doesn't distribute software but uses it solely for its own in-house purposes, the issues of modification and distribution are not issues of concern (except for the 1099 labor and subsidiary issues identified above). However, it is an issue to consider for Developers of software that works with open source software. The extent to which this is a risk worth taking may be a function of how the Developer's software actually interacts with the open source software.

Handling Patent (and Other) IP Infringement Issues

The GPL handles potential claims of patent or other infringement in an interesting way. Section 7 of the GPL states what happens if a court ruling or anything else happens that prevents a licensee from conforming to the GPL. An example of this occurrence would be if a court rules that a portion of the open source software is subject to a patent right, thereby precluding the right to distribute it for free. In that case, the licensee is simply precluded from distributing the offending code. The licensee is still required to conform to the GPL. This language is an attempt by the author of the GPL to keep open source software free from intellectual property claims.[27]

Handling Certain Countries' Contrary Laws

Section 8 of the GPL is similar to Section 7, except it deals with country laws that may be inconsistent with the GPL. In that situation, the original copyright holder under the GPL may add restrictions that prevent use of the open source software in that particular country.

Other Provisions

The GPL contains other provisions as well. There are provisions for periodic updates to the GPL itself. There are provisions for how one

---

[26] Short phrases, which API calls to other software normally are, are afforded relatively less protection under copyright laws, as are phrases that are limited in the number of different ways they can be actually expressed (e.g., how many ways can you say "open a file"?). In addition, U.S. courts have given some leeway to the actual use of expressive (i.e., copyrighted) elements of software when they are used solely to provide interoperability. In the European Union, reverse engineering for purposes of interoperability is specifically allowed, subject to certain limits.

[27] This hasn't worked perfectly, as can be seen from recent litigation brought by SCO.

incorporates portions of an open source software program into free versions of other software but that are not subject to the GPL. There are also disclaimers of warranties and limitations of liability. See the general discussion above for analysis of those provisions.

Conclusions

If you are an End User that wants to use unmodified open source software for internal purposes, the GPL gives you very little, if anything, to worry about. If an End User wants to modify the open source software and use it internally, then it may worry about the issue of 1099 labor, subsidiary use, and the risks associated with interoperability with the open source software. However, many End Users will examine these risks and determine that the gains from using the open source software are worth the business risks. For those organizations who embrace the open source software movement and actually contribute their modifications to the open source movement, the risk is even less.

Developers who want to use the open source software licensed under the GPL should carefully consider the risks associated with interoperability, as well as whether the GPL imposes restrictions on the Developer's proprietary code as a "collective work" of the open source software. They should specifically and technically examine how their proprietary software works with the open source software. Many will also decide that the use of open source software outweighs the risks; however, because the risks are greater, some are likely to stay away from open source software.

Lesser General Public License (LGPL) (Version 2.1)

General Conclusions

Under some circumstances, the LGPL allows the open source software to be used with proprietary software, specifically by linking,[28] without requiring the proprietary software to be licensed under the LGPL. This may allow software licensed under the LGPL to be used by End Users and Developers without giving up any proprietary rights in the code. Modifications to the software licensed under the LGPL itself are still governed by provisions very similar to the GPL.

The LGPL Generally

The Lesser General Public License[29] is probably the most technically complicated open source license. The reason for this complexity is its subject matter: the use of libraries by software programs. The LGPL arose

[28] Linking occurs as a part of the conversion between source code, written by programmers, and executable code, useable by computers. Linking takes various separate components of the software, such as modules and libraries of subroutines, and converts them into an executable program.

[29] In earlier versions of the LGPL, it was know as the Library General Public License.

because of special considerations related to the use of open source libraries that are linked with software that is not intended to be licensed under the GPL.

In its Preamble, the LGPL points out that if one takes software and links it to a library[30], the combined work is legally a derivative work of the library.[31] Accordingly, using a library that is subject to the GPL would render the entire work subject to the GPL.

There are occasions where having the combined work be subject to the GPL is not in the best interest of the free software movement. First, the free software movement may want the freeware version of a library or routine to be an industry standard. In this situation, the freeware versions will be more widely adopted if it is available for use in proprietary software as well as open source software. Second, there may be proprietary versions of the routines in the library that are readily available. In that situation, requiring any software using the open source routines to be licensed under the GPL reduces the likelihood that the open source routines will be used, which also seems contrary to the intention of the open source movement.

The most important distinction to understand when reviewing the LGPL is the difference between a "work based on the Library" and a "work that uses the Library." Generally speaking, a work based on the Library is a work that modifies the library itself. A work that uses the library is a software program that is linked with the library, normally during compilation.

Modifications to the Library Itself

A library that is licensed under the LGPL is generally treated in the same manner as if it were licensed under the GPL. If you distribute the unmodified library, you must license it in accordance with the LGPL. If you distribute modifications to the library, then those modifications must also be distributed in accordance with the LGPL, which requirements are virtually identical to the GPL, except that there are some special considerations associated with the nature of libraries that must be considered.[32]

---

[30] A library is a set of software routines or functions that can be used by calling them rather than writing that functionality into the software itself. Typically, these routines are "linked" into a software program when it is compiled.

[31] It is also true that the combined work is a derivative work of the software, but being a derivative work of the library is what causes the issue with the GPL.

[32] Section 2.d) of the LGPL provides: "If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful."

Use of the Library

As stated above, a software program that doesn't contain the library still is a derivative work of the library when it is linked and compiled with it. Portions of the LGPL related to this issue are less than completely clear, principally regarding the uses only of header information of the library and uses that entail a small amount of use of the library.[33]

Section 6 of the LGPL is the relaxation of the normal restriction on full and free distribution of derivative works. Section 6 contains the difference between the GPL and the LGPL.

As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work *under terms of your choice*, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.[34]

This means that as long as you allow a customer to reverse engineer for debugging and modify the software for internal use, you can charge a fee for the software and limit its subsequent distribution. The rights to reverse engineer and modify the software for internal use are typically not granted under commercial software licenses; however, even with these limits a Developer may want to consider using an open source library in its products.

There are some additional requirements, including the following:

- Identifying that the library is being used;
- Placement of copyright notices;
- Providing source code for the library;
- Providing a means to link the library;

---

33 The confusing language is as follows, with certain editorial references in brackets: "When a 'work that uses the Library' uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant [significant for what?] if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law [Why is this sentence necessary?].
If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)
Otherwise, if the work is a derivative of the Library [language probably incorrect, as the prior paragraph seems to give rights even if the work is a derivative work], you may distribute the object code for the work under the terms of Section 6. Any executables containing that work [not artfully stated] also fall under Section 6, whether or not they are linked directly with the Library itself.

34 Emphasis supplied.

- If not provided, an offer to provide the source code of the library; and

- Providing certain utility programs needed to reproduce the executable.

Other Provisions

The LGPL also provides a means for, and restrictions regarding, use of the LGPL library side-by-side with proprietary libraries. Patent and country-specific issues are handled in the same manner as the GPL. Just like the GPL, there are provisions for adopting newer versions of the LGPL as they come out, instructions for how to obtain exceptions the LGPL, disclaimers of warranty, and limitations of liability.

Conclusions

Once again, End Users aren't really impacted by the language of the LGPL, other than the 1099 labor and subsidiary questions in connection with linking or modifying the libraries that are subject to the LGPL. Many companies will find this as an acceptable risk. Developers who use libraries subject to the LGPL need to decide whether they can live with the requirements of Section 6, most notably the requirement that the Developer must grant its licensees the right to modify the code for internal use and reverse engineer for debugging.

Berkeley Software Distribution (BSD)License

The BSD license arose out of the University of California at Berkeley. The approach taken by the BSD license is much simpler than those taken by the GPL and the LGPL. It is also less protective of the open source movement and more flexible toward End Users and Developers. The BSD license consists of the following provisions:

- A copyright notice;

- Requirements that distribution must be accompanied with the copyright notice, the conditions in the BSD license and the disclaimers, with some requirements as to where those provisions are placed;

- A prohibition against using the contributors or affiliated organizations to endorse or promote the products;

- A disclaimer of warranties;

- A limitation of damages.

That's it. As you can see, the BSD license gives the software away, but it does not prevent someone from incorporating the software into a proprietary product.

For both the End User and Developer, the use of software subject to the BSD license does not expose it to the risk that the proprietary product developed using it will result in the inability to protect and defend its intellectual property rights in the proprietary product.

The MIT License

The MIT license is very similar to the BSD license. The use of software subject to the MIT license requires a copyright notice, a disclaimer of warranties, and a limitation of liability. The software license is otherwise unrestricted.

Apache Software License v2.0

In its market niche, Web server software, Apache is the most successful open source software of all, even more than Linux. In January 2004, Apache software was used in 67% of the approximately 46 million servers that make Web pages available on the Internet.[35]

Apache takes a different approach to open source contributions. Instead of using the concept of derivative works and copyright law to require contributions to the open source community, as with the GPL and LGPL, the Apache license provides for the voluntary designation of contributions to the open source software. Licensees are therefore given the option whether to create proprietary versions of Apache or contribute modified code to the Apache open source community.

Other features of the license are as follows:

- The Apache license handles the subsidiary issue. Subsidiaries are allowed to use the software as if they are the licensee.
- Linking is addressed by not covering it under the Apache license and in a manner suggesting that linking is allowed.
- There are separate copyright and patent licenses, which is a little unusual even for proprietary license agreements. The patent license only licenses those patents that would otherwise be infringed by the particular licensor's contributions or that would be infringed by the combination of the licensor's contributions and the other parts of the Apache software. This makes it more legally comfortable for contributors to contribute.
- If licensee institutes a claim of infringement, even as a defense, then license agreement terminates.
- Adding to the software *requires* attribution. This may help if infringement claims occur.

[35] See http://news.com.com/2100-7344-5139511.html.

- Unlike other disclaimers of warranty in open source licenses, implied warranties of title and noninfringement are also disclaimed.

- Contributors can add additional license terms. This means that a contributor can modify the code and then sell it as a proprietary package.

The Apache license is well-constructed, relatively easy to read, and provides a different approach to open source licensing from other open source software licenses.

Infringement Liability

The laws regarding patent and copyright apply to open source software just as they apply to proprietary software. As mentioned above, copyright infringement occurs when someone violates the exclusive rights of a copyright holder, usually by making copies, making derivative works, or by distributing someone else's copyrighted work. Patent infringement occurs when someone makes, uses, or sells something that is a machine, method, or process patented by someone else.[36, 37] In both patent and copyright claims, there is no requirement that the alleged infringer actually have knowledge of the copyright or patent or the infringement itself.[38] Accordingly, copyright or patent infringement claims can typically be brought against downstream users of the technology, such as End Users and Developers.

The analysis of potential infringement liability below examines whether there are any differences between proprietary software developed and licensed by a single software company or open source software on the following key issues: (i) the likelihood that the software created will contain information that could cause an infringement, (ii) the likelihood that the infringed-upon entity will discover the infringement, (iii) the likelihood that an infringement lawsuit would be brought against an End User or Developer, and (iv) the likelihood that the software developer will undertake the defense and indemnify for any damages suffered by the End User or Developer.

The true question in this situation is whether the infringement risks to an End User and Developer are different between open source software and proprietary software. If the risks are huge, but quantitatively and qualitatively the same, then there's no reason to choose one over the other due to the risk. The other question is how much is it worth, in dollars, to your company to get an infringement indemnity?

---

[36] Both the descriptions of copyright and patent infringement have been simplified to avoid unnecessary detail.

[37] Trade secret misappropriation is another possible claim; however, trade secret misappropriation claims against an End User or Developer would normally require a showing that the End User or Developer knew or had reason to know that the information was obtained inappropriately. Because this situation would not normally be the case in an open source context, trade secret misappropriation is not addressed in this white paper.

[38] Knowledge of the infringement can be quite important in the amount of damages awarded, however.

Chances of Infringing Materials Being Involved

Open source software is frequently developed by multiple persons who work for multiple employers, who provide code on a more or less anonymous basis (at least to the End User or Developer), who are relatively judgment proof, who provide the code for free, who at least partially subscribe to the notion that software must be free, and who know that they are providing it without any infringement warranty. Software companies, especially large ones, typically have procedures in place to reduce the risk that another's code gets into the software company's proprietary code. On balance, it is more likely that open source software will have copyright problems than a software company's proprietary software.

The above scenario is not true for all open source software providers. Some open source software is developed using fewer resources and with tighter controls. Others started from an open source foundation but then took more proprietary control of the code, even if it is still licensed for free. Accordingly, it is important to examine how the particular open source software was and continues to be developed.

While similar, patent infringement risks are less of a contrast between the open source community and purveyors of proprietary software. Some patent risks are simply unknowable, because the patent applications are not disclosed to the public for a period of time. Accordingly, there can be a future patent infringement that does not appear until after the code has been created, even assuming that the software company did a full patent search. In addition, many software companies don't do full patent searches. Even if the information is available, patents may still exist that cause problems. Therefore, while there is likely to be some difference between the patent infringement risks of using proprietary software and open source software, it is not necessarily that great a difference.

Risk of Getting Caught

In those situations where source code is available, it can be easier to determine whether someone has infringed on your copyright or patent. Making only object code available tends to hide some sins. Since source code is usually available with open source software and usually not available for proprietary software, the risks of getting caught can be greater with open source software.

Likelihood of Action Against End User or Developer

It is hard to talk about infringement risks without addressing SCO v. IBM. As a part of SCO's strategy, SCO has asserted claims against some End Users and has made inquiries against others regarding their use of Linux, seeking to obtain royalties on the use of Linux. One may contrast that to the multiple patent claims made against Microsoft products, where at least to the author's knowledge no End User has been sued or paid anything to resolve such claims. SCO v. IBM suggests that End Users using open source software are more likely to be sued than End Users licensing proprietary software,

at least for very large End Users.[39] It is less clear that smaller End Users and Developers are exposed to the same level of risk as very large End Users.

If a claim of copyright or patent infringement is available, it stands to reason that there is a greater risk of a claim being made with the use of open source software. The reasons are that the actual infringer is harder to find and more likely to be judgment proof. Why sue an End User when Microsoft has billions in the bank?

Some companies view the open source movement as a threat to the market position of their proprietary products. This is most evident in the relationship between Linux and Microsoft's Windows line of products. In 2003, Microsoft paid SCO a $16.6 million license fee.[40] Some sources believe that Microsoft is seeking to provide monetary assistance to SCO in order for SCO to pursue its litigation against IBM and End Users. If there are powerful companies who have a vested interest in seeing open source software fail in the marketplace, the risk of intellectual property claims is increased.

Who Will Defend and Indemnify?

In some open source situations, there is no one party that realistically can defend an infringement claim or pay any judgments that arise. Accordingly, the risk of not having a party to defend and indemnify is greater for open source software than with proprietary software. However, it is easy to overstate this risk. On average, patent infringement lawsuits cost about $3 million just to defend.[41] It takes a software company of some size to absorb this kind of litigation. In addition, the largest software companies don't always fully indemnify for infringement risk, meaning that an End User or Developer may still be left with the responsibility to defend and pay any settlements and judgments. As with all indemnifications, indemnities are only valuable to the extent that a party is willing to give one and is able to pay when the time comes.

Also, some open source providers will provide indemnifications for the products. Some are large enough that they will insist on resolving an infringement claim, knowing that the sale of new licenses and services will be impacted if the claims are not resolved. Therefore, whether there will be one or more entities who will defend an infringement claim and make it go away will vary depending on the open source developers involved, how the particular open source software is developed and maintained, and how much an economic stake the open source vendors have in the success of the open source software in question.

---

[39] Letters from SCO to large End Users were apparently sent to the Fortune 1000 and the Global 500. See http://en.wikipedia.org/wiki/SCO_v._IBM_Linux_lawsuit.

[40] See www.computerworld.com/softwaretopics/os/linux/story/0,10801,91145,00.html.

[41] AIPLA Report of Economic Survey 2001, Median of Estimated of Total Cost, Through End of Discovery and Inclusive, in a Patent Infringement Suit, with more than $25 Million at Stake, Table 22, p. 85.

SCO v. IBM

There are lessons to be learned in the SCO v. IBM[42] case and the related cases involving Novell, DaimlerChrysler, and AutoZone. Unfortunately, the claims made by the various parties against each other are both numerous and diverse. In addition, some of the most important claims have not been publicly fleshed out. Currently, it is not clear how good a case each party has against the other.

There is one strategy in these cases that is important to consider in connection with an End User or Developer examining its own risks, however. Because copyright law only protects the way ideas are expressed rather than the ideas themselves, there is no code in Linux that can't be rewritten to avoid an ongoing copyright claim. From the claims made, the estimates of the number of lines of code that are infringing range from a few hundred to a few million.[43]

Suppose for a moment that there are one million lines of offending code. If the open source community knew which lines of code were offending, how long would it take them to rewrite those lines of code? As already noted, it was estimated that thousands of programmers participated in the development of Linux. From that estimate, let's suppose that the open source community can marshal five thousand programmers to work on the project, a conservative number. This means that each programmer would need to write an average of 200 lines of code to fix the problem. How long would that take? Of course, it isn't nearly that simple. The task wouldn't break down into equal parts, and there's plenty of testing, vetting of code and the like that would need to be done. The point is, however, that even if it's a million lines of infringing code, once the SCO litigation is considered credible by the open source community and the offending code is disclosed, it won't be long before a revised version of Linux is available that will be free of offending SCO code. This analysis points to a strength of the open source community: strength in numbers. This strength means that the risk of ongoing copyright infringement is lower than many may have otherwise thought.[44]

Monetizing the Risk; An Economic Analysis

Legal risks, such as the risk of future infringement claims, should be balanced against other economics. Suppose an End User is evaluating whether to license one software package versus another. All the business and legal terms are the same except for two factors: one software company won't indemnify for infringements at all but offers its software for $100,000 less. How much is a software indemnity worth to your company? If your company can save a few million dollars by using open source software over proprietary software, is it worth the clear economic savings to take the risk

---

[42] For a good summary of these cases, see http://en.wikipedia.org/wiki/SCO_v._IBM_Linux_lawsuit. Note that Wikipedia is an encyclopedia, but it is an *open source* encyclopedia. Accordingly, its rendition of the facts may not be completely without bias.

[43] The disparity of the estimated quantity of infringement is one of the reasons that the claims are so hard to evaluate.

[44] In addition, it shows that SCO's strategy should be to delay the disclosure of the alleged offending code for as long as possible.

of the possibility of being sued in the future for infringement? If you had to buy your way out of infringement trouble by paying for a license in the future, would you still come out ahead economically?

For most companies, this is the analysis that is most important. Many companies will decide that the cheaper cost of open source software justifies taking some increased risk on the infringement side. Others who are less risk tolerant may not. It may be helpful to go through the analysis outlined above before making any decisions, because different open source software will present different risk postures.

Other Issues

Control

Open source software is really easy to download and use. A software programmer can frequently identify open source software, download it, and start working with it in a matter of minutes. When comparing that speed with a normal corporate purchasing processes, it is easy to see why programmers like open source. In addition, the open source community frequently provides solutions to specific problems, provides reference information to the software's use, and generally supports the work of a programmer who uses open source software. It is easy to feel a part of a community when using open source software.

To avoid the irresistible call of the Sirens — and to resist the temptation to steer toward certain death — Odysseus placed wax in the ears of his crew and had himself tied to the mast. Similarly, some corporate control needs to be in place to manage the use of open source software. The basic principles of control are relatively straightforward:

- educate programmers as to when using open source software is appropriate and when not;
- identify when programmers are trying to use open source software;
- determine what applications or programming tasks the programmers want to use the open source software for;
- review the applicable open source license agreements;
- determine whether the uses are consistent with the limitations of the open source software license;
- assess the infringement and other risks of using open source software;
- decide whether to use the open source software or not;
- monitor or otherwise control whether the open source software is used for other projects; and

- monitor compliance with any policies developed to address any of the controls above.

While the basics of these controls are easy to state, controlling the use of open source software can be a significant endeavor, especially for large organizations and organizations with less of an orientation toward process. What these controls look like and how they are constructed will vary widely depending on the size, culture and organization of the company.

Import Restrictions

The U.S. government places certain restrictions on the import of products and services from rogue countries. For example, one cannot purchase products and services from Syria or Cuba. Because open source software can be developed by multiple persons on a world-wide basis, it is possible that the use of open source software is potentially a violation of U.S. import regulations? The answer is probably yes. However, the risk would seem to be low that U.S. import laws would be enforced to this level. This risk obviously goes up to the extent that the software was principally developed by programmers from rogue countries. The author is aware of one anecdotal example where a software company was not allowed to license its software to the U.S. Department of Homeland Security because it was principally developed in a rogue nation, for security reasons. This anecdote did not involve sanctions for violations of import restrictions; the deal just fell through.

Conclusion

Hopefully, this white paper has given you information that makes it easier for you to make an informed choice whether to use open source software and under what circumstances. As you can see, the risks from a legal standpoint are generally the following:

- 1099 labor,
- subsidiaries,
- having your own software treated as a derivative work or collective work of the open source software, and
- infringement liability.

Your risk posture is affected by whether you are an End User who uses unmodified open source software, an End User who uses modified open source software, or a Developer. The particular open source license will impact your risk, as will the way your proprietary software interfaces with the open source software. By balancing the risks under your circumstances, the risks under the particular open source license agreement, your comfort with these risks, and the economic differences between the available choices, you should be able to make an informed choice whether using open source software is right for you. In any event, however, you should seriously consider

implementing controls to monitor and manage the use of open source software in your company.