September 14, 2008

The Evolution of GPLv3 and Contributor Agreements in Open Source Software

George E Finney, *Southern Methodist University*

The Evolution of GPLv3 and Contributor Agreements in Open Source Software

By George Finney

Abstract

Innovation in the computer software industry over the past 15 years increased at a frantic pace thanks in part to the Open Source Software movement. This community of software developers uses legal methodologies to enforce rights and induce others to follow their lead and with openness has come more innovation. The Free Software Foundation's General Public License (GPL) in particular embodies this spirit and uses a kind of recursive philanthropy or an Open Source-only sandbox, which requires other developers to create their software under licenses with compatible philosophies.

How the propagation of these licenses effects innovation in the software industry is the subject of this paper. This inquiry will examine methods used in other industries that embody similar propagation devices from which subsequent Open Source licenses could benefit, proposing that Open Source must balance the risk Open Source companies face with the basic donative intent of the community to continue to offer a cohesive doctrine other developers will continue to follow.[1]

In light of the most recent version of the GPL, this paper examines the drafting and revision process the community undertook and will examine what compromises resulted. This process, while a groundbreaking step for the community, necessarily creates new issues to be resolved and may be more reactionary to current events than necessary to support the community's goals of greater innovation.

Outline

I. Introduction

Imagine an orchestra composed of volunteers from all over the world, each with their own instrument. Imagine further that each volunteer writes and plays his own part in the symphony, separated by hundreds of miles. Finally, imagine this orchestra compared favorably to the best orchestras in the world. While this may be hard to imagine in a practical sense, this is exactly what Open Source Software developers have done over the course of the last 20 years. Many scholars have proffered theories as to why and how this process could work on such a large scale. This study is distinguishable because it seeks to understand how the course of Open Source will change in the years to come. Through a process of software donation, which can be equated to a kind of recursive philanthropy, a large number of computer programs have been developed across multiple categories of computer software. It appears, however, that the recursive aspects of the donative process have never been fully challenged as being consistent with the goal of innovation or the stated philosophy underlying Open Source development. This paper will ask: "do the ends justify the means"?

There seems to be some tension in the Open Source world between the ideals expressed and the practice of enforcing licensing agreements.[2] This tension arises in the

[2] The most concise source for the ideals of the Open Source community comes from the Free Software Foundation (FSF) and the GNU Project, which define the four core freedoms that must be in place for the FSF to consider the software "free." Those freedoms are:

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and adapt it to your needs.
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

Free Software Foundation, Free Software Definition (2007), http://www.gnu.org/philosophy/free-sw.html. The original version appeared in the initial

Open Source community through its definition of propagation. Propagation in this context is when the use of the Open Source software requires the same or similar Open Source license be applied to the code that is newly developed based on the earlier work. Propagation can arise through derivative works in the context of copyright law where one developer adds to a piece of software. Propagation arises in patent law through the notion of improvements, when a developer adds a new feature to an existing patented program. Because not all Open Source licenses are the same in this regard, the issue of propagation becomes complex. The contribute-back provisions found in some Open Source licenses, requiring code developed under a particular license be contributed back to the community, present a similar issue. These licenses allow an individual to use the Open Source code however they desire, so long as any new work created based on the Open Source code is given back to the Open Source project for further use by the community.

Since Open Source builds on both copyright law and patent law, the more fundamental issue of whether Open Source conflicts with either of these doctrines must be considered. It is possible that Open Source creates a competing model to that of the established modes of intellectual property, using their own functions against them. This paper will take a different position, contending both intellectual property and Open Source should coexist as complementary methods of regulating property. In traditional intellectual property, organization comes in the form of government regulation; Open Source, however, creates a self-regulated commons. This commons is not without

---

bulletin of the GNU created by the FSF containing only the first two freedoms. Richard Stallman, What is the Free Software Foundation, Feb. 1986, http://www.gnu.org/bulletins/bull1.txt.

controversy due to the community's own developing notions of how the commons should be regulated.

Commentators pay close attention to how proprietary code has infiltrated the Open Source area. This issue came to a heated point in 2004 when The SCO Group filed suit against Novell, claiming that infringing code existed in the Linux kernel.[3] After three years of legal discovery, the court determined that only 326 lines of code in the Linux kernel were potentially infringing.[4] In the meantime, the contribution processes for many Open Source projects tightened in order to discover proprietary code more easily.[5] With the recent court decisions regarding the SCO lawsuit and the company's subsequent bankruptcy, this issue has fallen largely into the background.[6]

While one problem has diminished, several other issues have come to the foreground. A large concern since the inception of the GPL is the issue of software patents and their consequences for Open Source software.[7] Several Open Source software licenses such as the Mozilla Public License and the Apache Software license have addressed this problem with explicit patent clauses. In addition, the third version of

---

[3] Darl McBride, *Open Letter on Copyrights*, Dec. 4, 2003, http://www.sco.com/copyright.
[4] Charles Babcock, *IBM Argues SCO's Case Comes Down To 326 Lines Of Code*, March 19,2007, http://www.informationweek.com/story/showArticle.jhtml?articleID=198001720.
[5] The Free Software Foundation requires copyright assignments on all of its projects as a way of insuring that all recordkeeping and copyright registration requirements. Eben Moglen, *Why the FSF gets copyright assignments from contributors*, June 20, 2007, http://www.gnu.org/licenses/why-assign.html.
[6] Robert McMillan, *SCO Declares Bankruptcy*, Sept. 19, 2007, http://www.networkworld.com/news/2007/091907-sco-nasdaq.html.
[7] Version 2 of the GPL, created in 1991, in its preamble states that "any free program is threatened constantly by software patents". GPL Version 2, June, 1991, http://www.fsf.org/licensing/licenses/info/GPLv2.html.

the GPL attempts to address the issue of software patents as well.[8] While software patents present a significant issue for Open Source developers, a more basic problem still exists. The debate over what constitutes a derivative work and how a license propagates itself is more widespread than many other topics currently being discussed by the Open Source community. In addition, many software companies are concerned about the effects Open Source software may have on proprietary code because of the ambiguity of many Open Source licenses.

The development of the propagation issue within the several versions of most widely used Open Source license, the GNU Public License, is the focus of this paper. This discussion examines the latest iteration of the GPL, GPLv3, specifically regarding how it addresses the issue of propagation. Finally, this examination contemplates software patents in light of how propagation functions differently in the context of patent rights versus the context of copyright. In order to continue to offer a cohesive doctrine that others will follow, Open Source must grow to balance the basic donative intent of the community with the risk that Open Source companies face.

The Open Source definition, as promulgated by the Open Source Initiative, is silent on propagation and contribute-back provisions.[9] This silence leaves room for a broad spectrum of licensing provisions one must be aware of, particularly because a programmer who 'reuses' code or reverse engineers an algorithm from an Open Source project may not fully examine the specific provisions of a license before implementing that code in a proprietary project. Further, at least three types of rights effect software:

---

[8] Free Software Foundation, *Opinion on Covenant not to Assert Patent Claims*, Aug. 3, 2006, http://gplv3.fsf.org/covenant-not-to-assert-dd2.html.
[9] Open Source Initiative, *The Open Source Definition*, July 24, 2006, http://www.opensource.org/docs/definition.php.

copyright, patent, and contractual Rights. Each type of right will have a different impact on propagation.

Part II of this paper examines the foundations of ownership in Open Source projects. The fundamental question is: who owns the right to an Open Source project? Several projects have taken a variety of approaches to answering this question, including centralizing ownership of contributions using Contributor Agreements as an addendum to the general Open Source License chosen by the project. In addition, this paper will examine the issue of the work-for-hire doctrine in light of ownership. The Free Software Foundation (FSF) has instituted a policy of requiring developers to disclaim any work-for-hire ownership claims the developer's employer might have.

Part III of the paper examines the several revisions of the GPL, up to the current iteration, GPLv3. This examination starts with the 'viral' nature of the initial GPL licenses and how the Open Source community has reacted and adapted the licenses accordingly. The examination proceeds to the drafting process of the GPLv3 and to the multiple issues that arise out of the developing notions of how propagation issues should be treated.

Part IV of this paper considers the effects software patents have on Open Source propagation. While many large corporations have 'donated' patents to the Open Source movement, the present analysis discusses how these patents will address the concerns the Open Source community has over software patents and whether they provide any benefits to the Open Source Movement. Examining the GPLv3's patent compromise and the ultimate decision to go with a patent covenant not to assert patent claims will be contrasted with the FSF's initial position requiring patent assignment. An interesting

result of the propagation development of the latest version of the GPL has developed: the FSF now claims that, because of the propagation provisions in GPLv3, the Microsoft/Novell partnership initially created to provide limited patent protection to Novell SUSE customers will spread to all Linux users once Novell begins shipping GPLv3 code.[10] While the FSF has made some strong statements regarding Novell and GPLv3, the newest version of the GPL contains a grandfather clause that exempts this particular agreement. The GPLv3 only addresses the consequences to Novell peripherally, assuming that the effects of the patent coverage of GPLv3 section 11 will "necessarily be visited upon Novell."[11]

The paper concludes by observing that there are still significant steps remaining in the evolution of Open Source. The current environment of Open Source licenses is more reactive, in contrast with the aspirational beginnings of the movement. Reactiveness in this area is dangerous, with particular concern that certain interest groups within the community may fracture. The analysis concludes that Open Source must return to its beginnings, and consequently move away from this reactiveness, by balancing the basic donative intent of the community with the risk Open Source companies face.

II. Identifying Ownership Issues in Open Source

[10] See generally, http://www.cbronline.com/article_news.asp?guid=486516F4-A7CF-4C09-922A-1CB470AC073C, http://www.informationweek.com/showArticle.jhtml?articleID=201000334, http://www.google.com/search?q=microsoft+novell+open+source+coupons+and+gplv3&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a.

[11] GPL Version 3, Third Discussion Draft Rationale, 27, March 28, 2007, http://gplv3.fsf.org/gpl3-dd3-rationale.pdf.

The first issue that arises when analyzing propagation in an Open Source project is determining the breadth of the array of owners for a given project. As software projects evolve, multiple developers contribute to the broader endeavor. The rights for a project are defined first by copyright law and then by contract. Depending on the Open Source license the developers release the software under, the analysis of ownership may be clear. For example, the Apache Software License and the Mozilla Public License both provide an explicit chain of title so the rights for a specific project are better defined.[12] The GPL itself only peripherally addresses the chain of title issue. GPLv3 purports to address this issue with automatic licensing, rather than providing for explicit chain of title provisions.[13] Since the majority of Open Source projects fall under the GPL, this is a significant source of uncertainty.[14] To resolve this problem, some Open Source projects have created Contributor Agreements to better define how to control contributions to that specific project. Purely in terms of the goals of the Open Source initiative, whether these Contributor Agreements live up to either the Open Source Definition or the Free Software Definition is unclear. A Contributor Agreement that is too restrictive or too permissive might violate the terms of these definitions. As a result, it is possible that, although the project uses the GPL, the project will not be a truly Open Source project.

Contributor Agreements are highly controversial in the Open Source community. At least one key developer for the Open Office project has complained about the requirement that the developers of the project license their code directly to Sun so that the

---

[12] LAWRENCE ROSEN, OPEN SOURCE LICENSES, SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW, Prentice Hall (July 2004).
[13] GPLv3 Second Discussion Draft Rationale, 23, note 18, July 27, 2006, http://gplv3.fsf.org/gpl3-dd1to2-markup-rationale.pdf.
[14] See generally Freshmeat.net, http://freshmeat.net/stats/#license. As of September 29th, 2007, the GPL accounts for approximately 64% of all Open Source projects.

company can release a proprietary version of its product.[15] Many of the most successful Open Source projects require similar Contributor Agreements, including Apache, Fedora, and MySQL. With the exception of Apache, these projects all have the ability to make a profit at the center of their Contributor Agreement requirement. While making a profit is not necessarily at odds with the philosophy of Open Source, the more closely tied the contribution of code is to profit-making, the more controversial these agreements become. Nonetheless, a Contributor Agreement plays a key role in putting contributions to an Open Source project on firmer ground. The authors of the GPL believe their license is strictly based on copyright licensing and insist it not be treated as a contract.[16] The issue with the GPL, then, is that developers need to know whether individual contributors hold the copyright for a project, or if the individual contributor implicitly is donating their code to the project because of their contribution.

Two possible ownership models exist for an Open Source project: either the individual contributor owns the source code or an organization owns it. If each contributor retains his copyright to the source code, this plays an important role in how propagation will develop for the project. The individual copyright method creates a much more complex determination of how a particular piece of code can be taken and reused in a different project. For example, must the project contact the individual author

---

[15] See History of Calc Solver, http://kohei.us/2007/10/02/history-of-calc-solver/ (Oct. 2, 2007). This view was criticized by another member of the Open Source community, Charles H. Schultz, who explaining that this procedure has been accepted by a larger number of developers than it has been rejected by, and avoids "kernel Babylon" where hundreds of contributors make it impossible to change the license for a given project. See Charles Schulz, *Thank you Michael, but no, thank you*, Oct. 9, 2007, http://www.groklaw.net/article.php?story=20071008124053220#note1.

[16] Eben Moglen, *Free Software Matters: Enforcing the GPL*, Aug. 12, 2001, http://emoglen.law.columbia.edu/publications/lu-12.html.

to determine if their code can be dual licensed or, perhaps, commercially licensed? How would a lawsuit to enforce the author's rights function in a project with over a 1,000 contributors across multiple continents? Which nation's copyright law would apply? A license that uses clear chain-of-title provisions makes this determination more manageable.

Centralization of copyright ownership simplifies this issue, but centralization is a very controversial aspect of some Open Source Projects. Some critics believe centralization contradicts the Open Source Principles altogether. They argue the best way to protect the public domain is to have a centralized body that can assert its rights against infringers. One important aspect in protecting an Open Source project is ensuring unknown interests do not burden contributions. "Works for hire"[17] for example, where an individual may contribute to a project, but the individual's employer is the actual owner of the contribution. Because of an employment agreement or because the developer may have written the actual code at their job, their contributions might fall into the definition of a work-for-hire.[18] In order to fall under the definition of a work-for-hire project, the work must fit into one of the nine categories protected under the Copyright Act[19]. A written agreement must specifically categorize, in advance, what constitutes a work-for-hire.[20] Some Open Source developers view the potential conflict of work-for-

[17] 17 U.S.C. sec 101.
[18] In order to be considered a work for hire, the work needs to either be made in the scope of employment or fall under one of the nine categories listed in the Copyright Act.
[19] If software falls under the work for hire doctrine, it would need to fall under the collective work or compilations category.
[20] The GPL itself does not appear to contain an explicit provision that works be made under the work for hire doctrine. An Open Source project may accomplish this with a Contributor Agreement, however the legal complications that arise may create problems for the Open Source project. For example, a developer may contribute software for

hire contracts as a risk and choose to protect themselves from receiving tainted code by requiring a Contributor Agreement.[21]

Alternatively, if Open Source falls under a decentralized model, contribute-back functions can be positive for individual contributors because they also secure partial copyright ownership of the whole project by an individual or an organization. A large organization with many developers contributing to a project actually may have less risk when contemplating joining an Open Source project because they could retain a large percentage of ownership in the project very quickly.

While Open Source licenses and projects provide both centralized and decentralized ownership of particular projects, there is another way to view ownership within the Open Source field: openness itself. While it seems ironic, some Open Source licenses provide greater freedoms than others. Completely liberal licenses like the BSD and MIT allow for a complete dedication to the public domain.[22] These licenses provide that anyone can use a piece of software for any purpose whatsoever. The GPL, in contrast, creates a GPL-only sandbox within the playground of the public domain. "Castles" that are built in the GPL sandbox must stay there. Developers can take some the tools they might use to build these "castles" in and out of the sandbox. These

---

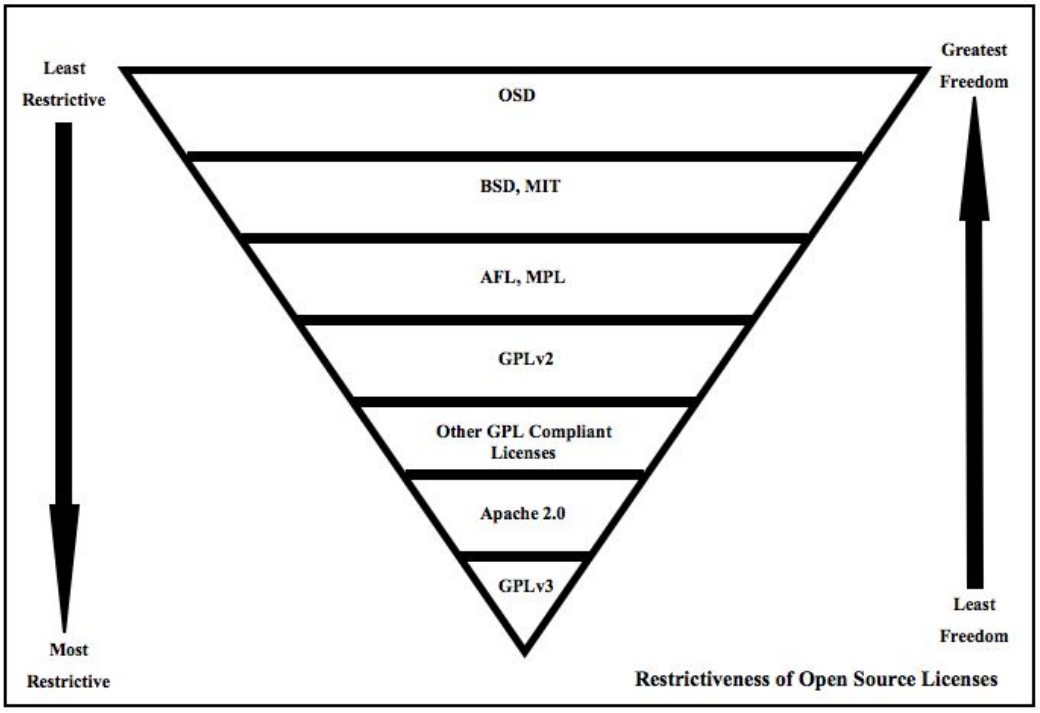
which he has a patent. While the particular license for the project may or may not have a patent provision (non-GPLv3 for example), the owner of the patent would still need to record the assignment with the USPTO.

[21] One Open Source project, Jive, expressed the work for hire concern publicly. Online interview with attorney Dan Ravicher by Slashdot community (June 5, 2001) http://interviews.slashdot.org/article.pl?sid=01/06/05/122240. Jive uses a dual licensing model and now requires a Contributor Agreement.

[22] BSD and MIT refer to the open source licenses created by the University of Berkeley or the "Berkeley Software Distribution" and Massachusetts Institute of Technology. Wikipedia article on BSD license, http://en.wikipedia.org/wiki/BSD_license (last visited Dec.13, 2007). Wikipedia article on MIT License, http://en.wikipedia.org/wiki/MIT_License

restrictions provide a model from which we can better understand the ownership rights in Open Source projects.

Figure A



[23]

Figure A illustrates the restrictiveness of several Open Source licenses. The Open Source Definition (OSD) represents the least restrictive and most free tier. The lower tiers represent the increasing restrictiveness and decreasing freedom of each Open Source license.

Importantly, not all Open Source licenses compatible with one version of the GPL are compatible with the other. For example, the Apache 2.0 license is compatible with

[23] Legal scholars differ on whether certain licenses are compatible with the GPL. This chart uses the list of compatible licenses provided by the FSF and the GNU organizations. For a complete list of GPL compatible licenses, see Free Software Foundation Compatible Licenses, http://www.gnu.org/philosophy/license-list.html (last visited Dec. 15, 2007).

the GPLv3 but not with GPLv2.[24] Taken in this light, the GPLv3 is not a subset of the GPLv2; instead, GPLv3 only intersects GPLv2 narrowly where some of the more basic provisions are shared. Additionally, while one license may be compatible with another license, the reverse may not be true.[25]

Copyright ownership in an Open Source project can carry significant implications. First, ownership gives rise to a cause of action in the event of a breach of license. More importantly, with regard to propagation, ownership provides the ability to change licenses. Consequently, ownership makes dual licensing[26] possible, which can be very lucrative for an Open Source company.[27]

What an Open Source license does not cover is also important to ownership of the underlying copyrights in an Open Source project and how or if the project is commercialized. Open Source reserves some of the grants that go along with copyright, like public display. Notably, Open Source does not cover some essential aspects of the code. Copyright protection does not extend to the ideas, algorithms, processes, or systems that underlie the concepts. Copyright protection does not extend to non-derivative works or compilations. Additionally, Open Source offers no warranties or guarantees concerning the software.

---

[24] Free Software Foundation, *FSF Releases "Last Call" Draft of GPLv3, May 31*, 2007, http://www.fsf.org/news/gpl3dd4-released.

[25] A developer may combine software licensed under the BSD license, for example, with software licensed under the GPL, however the reverse is not true. A developer may not combine code licensed under the GPL with software licensed under the BSD.

[26] Dual licensing refers to the practice of offering software under both a commercial license and an open source license. The commercial license will usually offer additional warranties and support.

[27] MySQL is the biggest example of this, attracting many of the world's best-known venture capital firms for funding. MySQL, http://www.mysql.com/company/investors.html (last visited Dec. 15, 2007).

Copyright protection does extend beyond the literal code of a program, but the determination of the degree of protection requires a fact-specific analysis. Copyright would protect a literal translation from one programming language to another, provided there are not any creative means employed in the translation.[28] On the other hand, if creative means were involved, this creative translation would be a derivative work and consequently would be required to comply with the licensing provisions. Copyright protection provides greater coverage for protection of the code, including the structure and architecture of the program.

The fundamental irony with propagation in Open Source software is this: because the public is free to look at the underlying code, software engineers have the ability to borrow ideas from Open Source software.[29] For example, one can learn an algorithm for a particular method of searching for text on a hard drive contained in an Open Source project based on the GPL. As stated above, copyright law does not protect the algorithm itself. If a developer simply removes the algorithm and rewrites it to fit into the architecture and structure of another operating system, for example, then the developer circumvents the GPL and no attribution would be required. This act of copying the algorithm creates a liability for developers attempting to learn from the code or companies attempting to accelerate the development curve for a project. When a developer copies more than just the algorithm, the danger of infringing copyright protections lurks. This danger is particularly important in the Open Source context where

[28] The definitions section of the third version of the GPL states that a "modified" work includes versions which have been translated. See GPLv3 Second Discussion Draft Rationale, note 8, page 4.

[29] See Freedom 1 of the Free Software Definition promulgated by the Free Software Foundation. The definition provides that the public should have "the freedom to study how the program works, and adapt it to your needs."

a commercial vendor runs the risk of having to apply an Open Source license to its proprietary code.

That one can circumvent propagation of an Open Source license begs the question whether propagation is an effective means of expanding the universe of software available under Open Source licensing. Some believe Open Source exists because of the need for development of software and the ease with which Open Source methods reduce transaction costs.[30] The history of Open Source software development over the last two decades suggests propagation has been an extremely successful method of expanding the software universe. Plausibly, the software universe that created under this model is a finite one. Once the universe reaches its limit, the number of new Open Source projects will dwindle. Similarly plausible is the potential that only a finite number of developers are willing to contribute under an Open Source license. Once the number of developers that are willing to contribute allocate themselves between the increasingly high numbers of Open Source projects, the development of new Open Source projects may diminish significantly.

III. The Evolution of Propagation in the GPL

As stated above, propagation has been a central theme in the development of the GPL. This is especially true in relation to the GPL's creation of the idea of a "sandbox" specifically intended to compel developers to use the GPL to build off other GPL code. GPLv2 did not use the term propagation itself; instead, the provisions in section 2 of GPLv2 are informally known as "viral." The GPLv3 updates this concept and explicitly

[30] Foremost among these is Yochai Benkler, who outlines his theories in Coase's Penguin and the Wealth of Networks.

defines propagation as doing "anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy."[31] With this definition, GPLv3 complicates its own interpretation. Through its attempts at internalizing language, its reactiveness to the Novell/Microsoft agreement, and its broader attempts to address software patents, the GPLv3 introduces a degree of uncertainty.

The GPL stands at the center of the debate in the Open Source community as to whether Open Source is anti-intellectual property or if it is just another complementary method that serves the same purpose of creating innovation and creativity. If Open Source is a competing method of organizing property or an alternative to intellectual property laws, then the GPLv3 becomes problematic. First, as demonstrated above, the GPLv3 is more restrictive than previous versions and many other alternative licenses. Being more restrictive reduces the GPL's ability to offer a full alternative to the private ownership rights of intellectual property law. While the new version of the GPL increases the repercussions of violating the license, its increased restrictiveness undoubtedly will lead to a decrease in its use. In addition, under the competing method theory, the Contributor Agreements software developers are required to sign are outright hypocrisy. Substituting a government regulated intellectual property regime that grants a monopoly for another requiring developers to donate their property, which effectively grants a monopoly to some third party, all in the name of freedom, is inconsistent at best.

The alternative is that Open Source is a method of regulating property, complementary to intellectual property law. Open Source, in this case, creates a parallel

[31] GPLv3 section 0, June 29, 2007, http://www.gnu.org/licenses/gpl-3.0.html.

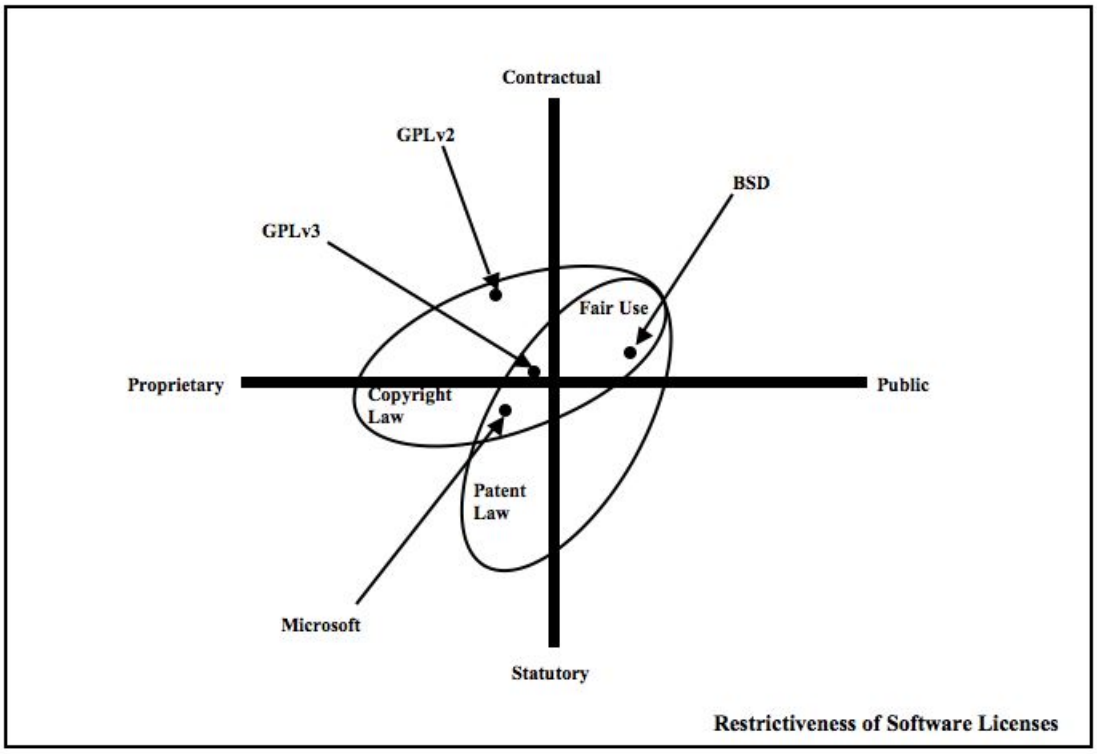
structure alongside existing intellectual property laws. In this case, the GPLv3 seems to attempt to create a sharper definition of Open Source. While creating consequences for Tivoization[32], anti-cross licensing agreements[33], and discouraging software patents, the GPLv3 may be better read as simply attempting to define more clearly the parallels while understanding inconsistencies between the two regimes.

The GPLv3 creates an intersection between that which is proprietary and that which is public, between the contractual and the statutory. As pictured in figure B below, the GPLv3 combines both proprietary and public components, taking elements of copyright and using contract or licensing to build a schema upon which it expands software development:

Figure B

[32] Tivoization refers to the process of building a platform around open source software, while preventing consumers from being able to modify the underlying software. Developers coined this term based on the popular Tivo digital video recorders. Tivo and the cable providers that use the software are reluctant to allow consumers to modify their boxes because of the support issues it would create if consumers inadvertently created problems with their equipment.

[33] The foremost among these cross licensing arrangements is the Novell/Microsoft agreement, which provides that neither company will pursue legal action against the other's customers based on software that infringes the intellectual property rights of the other company.



The GPLv3, like the GPLv2 is based heavily on copyright law. To a large degree, every version of the GPL has been a seemingly contractual device; however, the FSF has been staunchly opposed to the GPL's classification as a contract rather than a license.[34] The GPLv3 also incorporates some language dedicated to patent law. The GPLv3 falls short of actually incorporating patent law into the licensing language and, instead, creates a covenant not to file suit.[35] While not incorporating patent law, these provisions dealing with the patent issue raise further concerns that this version of the GPL is actually a

[34] Professor Rosen argues that this is largely for legal reasons. If a court determined the GPL was a contract rather than a license, the validity of the GPL would be placed in doubt through issues of consideration. Rosen, page 57-66.
[35] Free Software Foundation, *Covenant Not To Assert Patent Claims*, Aug. 3, 2006, http://gplv3.fsf.org/covenant-not-to-assert-dd2.html.

contract instead of a license and will complicate the legal issues concerning the GPLv3 in the future.

In version 2 of the GPL, propagation applies to derivative works. Take the example of a developer who creates a driver module that interacts with the Linux kernel, who distributes the code under the GPL. The driver is not part of the kernel itself, but it interacts with it. Under GPLv2, the module is not a derivative work and, therefore, the developer does not need to distribute the code under the GPL. This analysis becomes complex when considering other types of combinations of works. For example, is the result the same if the developer compiles the programs together? Is the result the same if the developer links two programs together?

Opponents of Open Source criticized the first version of the GPL, in particular, because of its viral nature. GPLv1 required the licensing of all programs associated with covered code. The language of GPLv1 requires a developer to distribute any work that "contains the program or any part thereof" under the GPL as well.[36] The second version of the GPL revises the requirement to be slightly more restrictive:

> 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.[37]
>
> You may not impose any further restrictions on the recipients' exercise of the rights granted herein.[38]

[36] Free Software Foundation, GPLv1 § 2, Feb. 1989, http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt.
[37] GPLv2 § 2[b]
[38] Id. at § 6

The second version of the GPL adds the statement that the developer of the work must license it under the GPL if it is *derived from* or contains GPL licensed code. The second version only slightly modifies the first version, which placed restrictions on the use of the work only when the program "contain[ed]" GPL licensed code. GPLv2 responded to the criticism of the "viral" nature of the first version of the GPL by including language that more clearly defined what would trigger the GPL propagation provisions. The trigger was to mirror the language of copyright law using the term "derivative."

In GPLv3, the restrictions on derivatives are less straightforward. The GPLv3 revises the language previously contained in section 2.b of GPLv2 and attempts to address the confusion surrounding propagation by defining what an aggregate work is:

> A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.[39]

Here, the GPLv3 is attempting to mirror more closely the definition of compilation for the purposes of copyright law, in which a compilation is a work formed by the collection and assembly of preexisting materials or data.[40] It light of other attempts at internationalization of the language of the GPLv3, compilations may have posed a problem, requiring a separate definition within the GPLv3 itself. Confusingly, GPLv3 does not define the term "combine" which makes interpreting this provision difficult. Does "combination" mean causing the two programs to (a) run together on the same

[39] GPLv3 § 5.
[40] 17 U.S.C. § 101.

computer at the same time, (b) compiled together, or (c) linked? Essentially, this confusion is the same issue as the problems discussed above with interpretation of the GPLv2 and the scope of propagation under that license.

The third version of GPL also introduces an explicit definition of the term "propagation."

> To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable Copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well. [41]

In addition, section 5 of GPLv3 completely revises the language concerning propagation of the license previously contained in section 2.[42] The term "derivative" is conspicuously absent.[43] In its place, GPLv3 attempts to create its own definition of a compilation at the end of section 5. Opponents of Open Source criticized previous versions of the GPL because the language that was sometimes inconsistent with the language of copyright law. The GPL may have been using the term "derivative" with a different meaning from that which the copyright laws intended. The GPLv3 does not address this concern. One possibility is that the drafters of the GPLv3 are attempting to broaden the language of the license so that it is more consistent with both copyright law and patent law.[44] The danger is that now the terms are too general and overly broad. If a developer challenged the license, an individual contributor might not have ownership of the rights GPLv3

---

[41] GPLv3 § 0.
[42] Id. at § 5
[43] Free Software Foundation, *Opinion on Denationalization of Terminology*, Aug. 3, 2006, http://gplv3.fsf.org/denationalization-dd2.html.
[44] I use the term "drafters" here not to single out any individual group, since the GPL revision process was very much a community effort.

contemplates him assigning. Under these circumstances, the GPLv3 demonstrates it is more consonant with Open Source as a complementary method rather than a competing method of regulation. As a complementary method of regulation, the GPLv3 regulates a larger sphere of rights, or, at the very least, unifies multiple statutory schemes for dealing with intellectual property rights.

The GPLv3 abandoned the "derivative" language after the second draft. The drafters completely removed the following section:

> To the extent that identifiable sections of the modified work, added by you, 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 convey them as separate works, not specifically for use in combination with the Program.[45]

The comments to the changes between the second discussion draft and the third discussion draft indicate this paragraph was deleted because it was unnecessary, and no specific explanation was given for why the term "derivative" was completely removed.[46] One explanation for this deletion is the attempt to "internationalize" the GPL, so that its language is more neutral. Internationalization, however, may not be the only cause for making the deletion. The underlying philosophy of creating a complementary regulatory framework may be at work here, also.

The FSF discusses the complexity of the internationalization process in specific detail in the second discussion draft. The first language change was a replacement of the definition of "modification" with language that better denationalizes the terms of the

[45] Free Software Foundation, GPLv3 second draft §5, July 27, 2006, http://gplv3.fsf.org/gpl-draft-2006-07-27.html.
[46] Id.

GPL.[47] The process of internationalizing the language of the GPL clearly is a difficult one. The comments to the GPL are littered with specific references to United States law, especially when referring to patents.[48] Ultimately, the question is whether the more general language is good for the GPL. In other words, would a U.S. court find that the new language is specific enough to fit into existing domestic copyright law?

The changes to the GPL seem to be an attempt to increase the amount of works covered by the license. It appears, internationalization attempts to broaden the meaning of what the drafters consider propagation to be within the GPL. The criticisms of the GPLv2 centered around the disparity between the GPL's use of the term "derivative" and the use intended by copyright law. In reality, this change in language may be an attempt to build the GPL sandbox even bigger.

Another change in the GPL accomplishes this goal: the newfound compatibility with other Open Source licenses found in Section 7 of GPLv3 makes the GPL sandbox even bigger. Section 7 contains language intended to make the GPLv3 more compatible with other Open Source licenses, most notably the Apache 2.0 license. Section 7 specifically covers any additional terms an Open Source project may consider adding and allows Open Source projects to reserve rights under trademark law. Section 7 permits developers to vary their disclaimers of warranty or to make different provisions for preservation of legal notices. Missing from section 7's discussion of additional terms are

[47] GPLv3 Second Discussion Draft Rationale, note 8.
[48] The GPLv3 Third Discussion Draft Rationale notes for example that in the user products definition, the drafters relied on the Magnuson-Moss Warranty Act, a federal consumer protection law in the United States. See Rationale section 1.2, page 10. In Section 3 of the GPLv3, the drafters specifically refer to the DMCA. See Rationale, section 2, page 14. In reference to Covenants, the drafters used US law as it primary reasoning for changing language related to patents. See Rationale section 3.3.2, page 20.

any terms that might be contained in a Contributor Agreement. Section 7 may also allow or ratify some Contributor Agreements, as discussed previously, although developers should consider each agreement on an individual basis. For example, if a Contributor Agreement effectively adds terms to a GPLv3 licensed work that are not contemplated in section 7, such an agreement, which may have been valid under GPLv2, may not be valid under GPLv3. Since the FSF uses its own Contributor Agreement, it appears Contributor Agreements are generally permissible; however, how far a company might be willing to take such an agreement is unclear. A Contributor Agreement might contemplate a blanket patent assignment for all patents received both in the past and in the future rather than a covenant not to sue, which goes further than the GPLv3 requires. Such an agreement adds an additional term to the GPLv3, seemingly in violation of GPLv3 §7.

Simply requiring Contributor Agreements to comply with the Open Source principles is ambiguous. One source of confusion in the Open Source initiative is whether the Contributor Agreements many Open Source projects are currently using are required to comply with the OSD. Since Contributor Agreements are outside the specifics of the Open Source license, which is OSI approved, a project may be able to effectively add terms that are inconsistent with Open Source principles. For example, if a project were to require developers sign different Contributor Agreements based on their country of origin, country of residence, employment status or primary language, this would violate the fifth principle of the Open Source Definition.[49] Such discrimination could be reasonable. For example, the project may need the Contributor Agreement to

[49] Section 5 of the Open Source Definition contains a non-discrimination clause.

comply with the laws of the country where the developer is located.[50] The Open Source Initiative only certifies licenses, it does not track individual projects using a license or whether projects add or remove additional terms. As Michael Tiemann[51] puts it, "The OSD is not a business plan." The group that creates a particular license must enforce it. In the case of the GPL, this enforcement falls to the FSF. The Apache Software Foundation enforces the Apache license. The FSF has been diligent in pursuing individual projects that violate the terms of the GPL outright. Because of the sheer number of GPL projects, enforcing compliance of Contributor Agreements for all projects is a huge undertaking. Requiring the Contributor Agreement provisions within the license itself is more reasonable.

IV. Propagation and Software Patents

Propagation in the software context becomes more complex when the developers of the GPL attempt to go outside what has been the foundation of their license since its inception: copyright law. The reality of technology law is that the drafters of the GPL must deal with the issue of software patents, but, in so doing, there are significant issues the GPL needs to address. GPLv3 is the first version of the GPL to explicitly provide

[50] While forking a project can circumvent Contributor Agreements, forking has many disadvantages, particularly to an individual developer. An individual developer would not have the ability to maintain the project on his own, and the bulk of the developers may remain with the project. This is true particularly where a discriminatory Contributor Agreement allows for different agreements for contributions sponsored by a corporation versus directly by an individual developer. Those developers protected by the corporate agreement would have no incentive to move to the forked version. In addition, as Danese Cooper of the Open Source Initiative points out, the fork is potentially vulnerable because it needs the support of the original copyright holder in the event the project is challenged.
[51] Michael Tiemann is currently the Vice President of Open Source Affairs at Red Hat, Inc., a vendor of the Linux operating system, and President of the Open Source Initiative.

how the license will handle patents. Since the GPL is so closely based on the rights provided by copyright law, previous versions have not fully addressed the additional complexity of how software patents will affect the license. Copyright law provides that improvements to a piece of software are derivative works. The same is not true in the realm of patent law. For the purposes of patent law, non-novel improvements require new patents. Furthermore, the assignments of patent rights require recording with the USPTO, while copyright law requires no such recording of assignments.

An additional incompatibility with Open Source software and software patents comes from patent law itself. While copyright law covers derivative works, improvements to a piece of software would require a new patent to cover the new software feature. Since software develops so rapidly, especially in the area of Open Source, requiring a new patent for each new feature presents a problem for Open Source. Requiring such frequent patents effectively slows the pace of development and creates more uncertainty as to the rights surrounding a new feature in an Open Source project. Since the discussion in the software community is concerned with the effect that existing software patents have on the development of Open Source software and with the potential minefield they create, this paper will only examine patents in the context of the issues raised by the GPL.

The second version of the GPL contained a reference to patents only in the preamble. The preamble contained a "freedom or death" clause which required that either software patents be licensed openly to everyone or not be licensed at all. This has led at least one legal scholar to conclude that the GPLv2's language is an express patent

grant for any software the developer has distributed under the GPL.[52] While the "freedom or death" clause represents an interesting footnote on the that the developers' awareness in 1991 for what the issues concerning software patents would be, most scholars agree the preamble to the GPL has no legal effect.

Software patents have been very controversial in the Open Source software community. Patent law creates a monopoly for the owner, which is antithetical to the principles of the Open Source community. Developers have created several Open Source licenses in the last decade that attempt to deal with the issue of software patents, but, at its core, patent law may not be reconcilable with the philosophy of Open Source. Patents on average can take 5 years to obtain and require investments reaching hundreds of thousands of dollars. Why any rational party would go to the time and expense of securing a patent, only to freely release it to the community as a whole is unclear. Many software companies have contributed their patents to software defense organizations, but whether these patents provide any greater protection than what might have been available otherwise debatable.[53]

The GPLv3 addresses the issue of patents in section 11:

> Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use,

[52] Terry J. Illari, Mass Licensing – Part 2: Open Source Software Licensing, 831 PLI/Pat 279, 295 (2005).

[53] These patent 'pledges' seem to come in the form of assertions that a company will not enforce their rights, and may not be patent assignments in the strict sense. The Linux Foundation's Patent Commons Project for example lists 530 patents, indemnification agreements, and other pledges not to sue. See generally, Patent Commons Project, http://www.patent-commons.org/resources/about_commitments.php (last visited Dec. 15, 2007). These agreements may not rise to the level of an actual assignment, and may not be enforceable at all.

> sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.[54]

This provision is problematic because the GPL may not be able to accomplish what it attempts to solve through the means of patent licensing. The assignment of a patent requires that the assigner record the assignment in order to be valid. The GPL does not address what happens if a developer contributes code to a GPLv3 project but fails to record the assignment of the patent license. Patent law provides specific recording requirements for the assignment of patents:

> An assignment, grant, or conveyance shall be void as against any subsequent purchaser or mortgagee for valuable consideration, without notice, unless it is recorded in the Patent and Trademark Office within three months from its date or prior to the date of such subsequent purchase or mortgage.[55]

The most dramatic change in the language during the GPLv3 revision process came between the first and second drafts. The first draft of GPLv3 contained a specific patent license grant, which would have created the issue discussed above. The final draft of the GPLv3 avoids this problem by defining a patent license to be an agreement not to enforce patent rights. This provision generated a good degree of concern on the part of the community for two reasons.[56] First, the impact on the patent holder was disproportionate to the act of merely distributing code.[57] The community feared patent holders would have to register every assignment with the patent office, possibly in multiple countries. The FSF rejects this argument on the moral ground that software

[54] GPLv3 § 11.
[55] 35 U.S.C. 261
[56] See Covenant Not to Assert Patent Claims.
[57] GPLv3 Third Discussion Draft Rationale, Section 3.3, page 17.

patents should not exist at all.[58] The second issue the community raised with regard to the patent license grant is that the due diligence required to review all distributed GPL-covered software is unreasonable.[59] To this argument, the FSF concedes that holding patent holders accountable to this requirement would mean large patent holding companies would likely choose to remove themselves from the GPL distribution process.[60] The drafters of the GPL conceded by changing this patent license grant into a covenant not to assert patent claims. The drafters of the GPL claim they made the change as a "partial concession" so that the conditions of section 10 would apply to the assertion of patents. The stated reasons aside, it would have been very difficult for the Open Source community to comply with a patent license grant, and it would have severely limited the adoption of the GPLv3.

In contrast to the desire to increase the distribution of GPL-covered software, some of the language in the GPLv3 creates an exception whereby a distributor can no longer distribute code. Section 11 of GPLv3 contains language intended to deal with the Microsoft-Novell pact.

> A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations

[58] Id. at 18.
[59] Id. at 17.
[60] Id, at 18.

> that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.[61]

The drafters intend this language to prevent discriminatory practices among Open Source distributors. The criticism of this language is that the Microsoft-Novell agreement enables the distribution of Linux and other pieces of software into organizations that had never considered implementing Linux before, Wal-Mart being the chief example.[62]

Section 11 also contains language directed specifically at Novell or other providers that might consider a similar arrangement. The agreement, as pointed out in the Novell discussion supra, was a non-exclusive agreement between the two parties. While the Microsoft-Novell agreement was grandfathered by the clause in Section 11, no subsequent agreements will be allowed under GPLv3, which effectively makes this agreement an exclusive one for Novell, as the lone Linux distributor to reach such an agreement. If, as the FSF argues, paragraph 6 of section 11 addresses the Novell Agreement, then it is unclear why paragraph 7 is necessary.

Paragraph 6 of Section 11 specifically pertains to the Novell agreement, and this reference is one of only eight times that the GPLv3 uses the term "propagate." This provision requires that if you propagate a GPLv3 covered work, while granting a patent license to a few users, you must grant the same patent license to all the recipients of the covered work and, in addition, works based on it.

> If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered

---

[61] GPLv3 § 11 paragraph 7.

[62] John Palfrey, Executive Director of the Berkman Center at Harvard Law School moderated a panel that included members of the original team from both Microsoft and Novell. The audio of the discussion is available at: http://www.peapodcast.com/msc-oss-sig/index.html#osssig-2007-09-26-18-00-48 (last visited Nov. 16, 2007).

> work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.[63]

The first issue with this language, as stated above, is the burden it places on the licensor. Registering patent assignments with the USPTO would be impractical; making this section is punitive. One factor that may soften this punishment is that the language may imply not that Novell or Microsoft must extend the license to all users of the software, but that they cannot discriminate among the individuals to whom they convey the software. A court might be unwilling to extend patent licensing to any party that ever downloaded a piece of GPLv3 software. It is much more likely that a court would grant a very narrow remedy. Furthermore, the language of section 11, paragraph 6 contradicts an earlier statement by the FSF in the initial discussion draft, stating it "does not entirely share the current enthusiasm of others in the free software community for including *broad* forms of patent retaliation."[64] The FSF may argue that the Novell clause of section 11 is not "broad" patent retaliation, but it is retaliation nonetheless, which makes this version of the GPL much more belligerent than previous ones.

V. Conclusion

The development of Open Source licensing is an ongoing effort. Despite some initial controversy, the GPLv3's adoption rate is increasing.[65] This raises the question of

---

[63] GPLv3 § 11.
[64] GPLv3 First Discussion Draft Rationale, section 1.2 page 3, Jan. 16, 2006, http://gplv3.fsf.org/gpl-rationale-2006-01-16.pdf (emphasis added).
[65] See GPLv3 Conversion Project http://gpl3.palamida.com:8080/index.jsp. As of September 28th, 2007, 734 projects have converted to the GPLv3. This means GPLv3 has taken approximately 1% of the market share of Open Source projects in 4 months. This

what the next version of the GPL will have in store. Potentially the biggest issue facing Open Source Developers is which Open Source license to choose. While the GPL may remain the most widely used license among developers, its market share has slowly declined over the past five years. Developers have followed the GPL model creating hundreds of Open Source licenses. Companies like Sun, Mozilla, Microsoft, and many others have crafted their own Open Source licenses, all of which meet different needs and concerns. These licenses all grapple with the issue of whether Open Source is fundamentally about creating a competing model of intellectual property or whether they are attempting to create a complimentary method of regulation which improves upon intellectual property's ability to foster innovation.

Ultimately, the biggest criticism of the GPL is that the license is very reactionary. The main changes that take place in the third version of the GPL are specific reactions to developments in patent law, particularly the spreading adoption of software patents. Furthermore, the GPL is reacting to conduct by specific companies. The commentary to the GPL drafting process coins the term "tivoization" in order to define the process of obfuscating the code to end users, a step that only few companies have taken. Further, much of the language around software patents has to do with a specific agreement between two companies, Novell and Microsoft.

---

puts the GPLv3 into 10th place in the top 10 Open Source licenses. See Freshmeat, License Breakdown, http://freshmeat.net/stats/#license (last visited Sept 28, 2007). In contrast, another survey released on September 25th, 2007 by Evans Data shows a different perspective. Surveying 380 Open Source developers, the study indicates that 66% of developers do not plan to implement the GPLv3 in the next year while another 43% say that they will never implement the GPLv3. Evans Data, *Open Source Developers Staying Away From GPLv3, New Evans Data Survey Shows*, Sept. 25, 2007, http://biz.yahoo.com/bw/070925/20070925006182.html?.v=1.

With the GPL being so specifically tailored to these relatively current issues, the question becomes, will a further revision be required when the next hot button issue arises? Doesn't this activity indicate that GPLv4 is close around the corner, given the current pace of the technology industry and the developing nature of technology law? The drafters of the GPL compare the process of developing the GPLv3 very favorably to a legislative process and claim it is representative of that for which a democracy should strive.[66] While this is a very noble goal, it subjects the GPL to the same criticisms of any legislation: that most legislation only happens in response to a perceived threat. The reactiveness of the GPL means once the next method of GPL circumvention not considered in the current version is discovered, there must be a modification to address it. While the GPL process was a good one, the Open Source community may not have the resources to constantly revise licenses.

Most likely, the next version of the GPL will address the issue of software patents very differently. The initial draft created by the FSF (not by the Open Source community as a whole) intended to force patent licensing without regard for the consequence it might create for the patent holder.[67] Because of their highly inflammatory rhetoric, it seems likely that they will reinsert the patent licensing issue into the discussion. Patent law is also subject to change and has evolved greatly over the past 20 years.

The ultimate goal of the GPL and the Open Source movement should not be to create a "sandbox." As a complementary method of regulation, the method begins to limit itself if it becomes exclusive and cannot grow using the entire sphere of property.

---

[66] Eben Moglen presented a lecture in Edinburgh, Scotland entitled "The Global Software Industry in Transformation: After GPLv3" the text of which is available at: http://www.archive.org/details/EbenMoglenLectureEdinburghJune2007text.

[67] GPLv3 Third Discussion Draft Rationale, section 3.3, page 17-18.

The "sandbox" concept can only to help foster the growth of free software during its infancy, similar to how one might protect a young plant with a net. Free software should ultimately be free, even from the control of the Free Software Foundation. While the abolition of software patents might be an important step in that process, the evolution of Open Source software is incomplete.

The ultimate goal of an Open Source License should be consistent with the values of the community and the core philosophy of the Free Software Foundation and Open Source definitions. The basic goal should be to foster creativity and innovation. This philosophy is similar to the goal of the copyright and patent systems. The differentiator between the two philosophies is Open Source's philanthropic element, while statutory schemes have focused on creating property rights. There is a thriving ecosystem around Open Source, but that ecosystem also depends on businesses that necessarily must use some form of commerce and, consequently, property rights in order to function. Open Source must balance these two competing spirits rather than rejecting a particular property scheme as immoral simply because it is commercial. Similar to the approach the GPLv3 takes with the denationalization of terminology, it seems plausible that a neutral approach remains possible on an issue like patent law, which may be amended or vary from country to country. To completely dismiss patent law is to abandon a mature method of fostering innovation for another that is still immature.