When Too Much Is Enough: Addressing the Rising Number of Open Source Software Licenses and their Effect on Innovation

Natalie G Banach

Available at: https://works.bepress.com/natalie_banach/1/
When Too Much is Enough

Addressing the Rising Number of Open Source Software Licenses and their Effect on Innovation

By Natalie Banach
William S. Richardson School of Law
University of Hawai‘i at Manoa
Second Year Seminar
Professor Jill Ramsfield
Spring 2010
Introduction

In the late 1970s, the average consumer would have paid around $900 for a commercial airline flight between New York and Paris and spent approximately seven hours on the plane en route. Now if the principle dictating the speed with which computer hardware advances\(^1\) was applied to the airline industry, today that same flight would cost about a penny and take less than one second.\(^2\) The speed of technological innovation illustrated here applies not only to the computer hardware industry but examples can be found throughout the software and Internet industries. These examples include the technological innovations behind the increase in speed and reliability of broadband Internet, as well as the advanced computer programs powering military operations, and even everyday iPhone applications. What is unseen in these examples of technological innovation however, is the role that the law plays in fostering innovation. Commentators and legal scholars often lament that the law is slow to change and cannot keep up with

---

1. This principle in hardware computing is referred to as Moore’s Law because it was first articulated by Gordon Moore, the cofounder of Intel Corporation. See ERIK BRYNJOLFSSON & ADAM SAUNDERS, WIRED FOR INNOVATION: HOW INFORMATION TECHNOLOGY IS RESHAPING THE ECONOMY 8 (The MIT Press, 2010) (“Gordon Moore predicted in 1965 that the number of transistors on memory microchips would double every year, and in 1975 he revised his prediction to every two years. What became known as Moore’s Law has held for more than 40 years as if the financial bubbles and busts never occurred.”).

advances in technology. But, this paper highlights the efforts of a group of software programmers who in fact used the law to overcome certain barriers to technological innovation through a movement centered on “free and open source software” [FOSS].

The FOSS movement is rooted in software creators’ desire to promote innovation in software through collaboration. This is an idea, which on a practical level, calls for an open exchange of software between FOSS participants by reworking the exclusive copying, modifying and distribution rights of copyright law that automatically attach when software code is written. The groundbreaking movement grew out of software programmers’ increasing frustrations with restrictions in modern copyright law.

Nevertheless, these same FOSS programmers now face another legal challenge as complications arise from the growing number of licenses being used to support FOSS applications. Licenses give permission “to commit some act[s] that would otherwise be unlawful” and they are the backbone of the FOSS

---

3 Frank H. Easterbrook, Cyberspace and the Law of the Horse, 1996 U. CHI. LEGAL F. 207 (1996), reprinted in PATRICIA L. BELLIA, PAUL S. BERMAN & DAVID G. POST, CYBERLAW: PROBLEMS OF POLICY AND JURISPRUDENCE IN THE INFORMATION AGE 18-19 (West Group 3d ed. 2007) (“I don’t know much about cyberspace; what I do know will be outdated in five years (if not five months!); and my predictions about the direction of change are worthless, making an effort to tailor the law to the subject futile.”).
5 BLACK'S LAW DICTIONARY 921(6th ed. 1990). See also supra note 4, at 364-65 (“By definition, a license is a contract granting permission to another party to perform some act. In a classic copyright context, a license may specifically be looked upon as a covenant not to sue for infringement of a right, and thus, a software license is a covenant not to sue for copyright infringement. The licensor is ostensibly the owner of the copyright and is permitting the downstream end-user licensee to violate one or more of the Section 106 Rights granted to the owner of a copyright. For example, the licensor is granting a right to the licensee to copy the work by installing a copy of the code or pieces of the code on a
movement because they allow programmers to grant privileges to downstream users that modern copyright law would otherwise restrict. This paper proposes that the solution to the proliferation of FOSS licenses lies in applying an easy-to-understand framework to specific license provisions. The application of such a framework will help license authors determine which FOSS licenses are in line with the movement’s desire to promote innovation, and eventually decrease the number of incompatible and complicated licenses that hamper innovation. Ultimately, the paper will demonstrate that the law can keep up with technological advances when legal practitioners and the developers of the technology work together to understand the interplay between law and technology.

Before analyzing the problem of license proliferation however, it is important to understand what FOSS is and how it functions. *Jacobsen v. Katzer* is a recently decided Federal Circuit Court of Appeals that offers perhaps the most eloquent summary of how free and open source projects operate:

Open source software projects invite computer programmers from around the world to view software code and make changes and improvements to it. Through such collaboration, software programs can often be written and debugged faster and at lower cost than if the copyright holder were required to do all of the work independently. In exchange and in consideration for this collaborative work, the copyright holder permits users to copy, modify and distribute the software code subject to conditions that serve to protect downstream users and to keep the code accessible.6

---

The thing that is “open” in FOSS is the computer source code, or the “collection of instructions a computer programmer writes to tell a computer what to do.”

This source code is made freely available so that downstream users can modify, improve and debug software without worrying about copyright infringement. In addition, although the software is often available free of charge or at substantially low prices, the “free” in FOSS more accurately refers to the right to access source code. In order to promote these principles and activities, FOSS programmers turned to the law.

From a legal standpoint, the ultimate power of the FOSS movement lies in its creative open licensing scheme that provides downstream users with the right to use, modify, distribute, and publish software programs under a “some rights reserved” license agreement, as opposed to the traditional “all rights reserved” copyright schema. Ultimately a free and open source “licensing philosophy” is used by owners of copyrighted subject matter to distribute material “in recognition of the axiom that collaboration is better than insular behavior.”

Perhaps a practical example is also illuminating: on the one hand, the Microsoft Windows operating system is proprietary technology; it is illegal and a

---

8 Id. at 187-188. See generally supra note 4.
11 See supra note 4.
violation of copyright law to burn a copy of the Microsoft Windows operating system and give it to a friend.\textsuperscript{12} Also, generally only Microsoft employees work on developing and improving the company’s software programs. On the other hand, FOSS projects such as the recently deployed Google Chrome web browser encourage the public to make adjustments, copies, and improvements.\textsuperscript{13} The goal of Google’s “Chromium” project is to “build a safer, faster, and more stable way for all Internet users to experience the web,” and the project allows users to make contributions to the source code under the permissive Berkeley Software Distribution (BSD) license, which represents a family of open licenses.\textsuperscript{14} The Google web browser is arguably more adept at promoting innovation because it fosters collaboration and improvement not just between its own programmers, but for every Internet user.

Thus, the refashioning of the law by FOSS participants demonstrates that the law can in fact change to meet new technological demands. Now these same software programmers, together with license authors, must play a role in solving the license proliferation problem. The increasing number of complex licenses has created an environment with many pitfalls that could ensnare users in a web of licensing litigation. Therefore, the end user should make sure they know what license provisions mean and, when creating new software programs using FOSS, they should be careful to pick only the licenses that are most relevant to their

\textsuperscript{14} Google, Chromium Project, http://code.google.com/chromium/. (The name of the web browser is Google Chrome, whereas the name of the project is Chromium.)
project. This paper offers an analytical framework for helping practitioners and software programmers achieve this understanding.

A Historical Perspective: The Ability of U.S. Copyright Law to Foster Technological Innovation

What is innovation? Innovation for the purposes of this paper should be conceived of as “the creation of the first copy of a good, process, or idea that did not exist before.” Innovation as the “engine of change” propels social and economic development. Promoting technological innovation is therefore of paramount interest and governments often intercede to support innovation through the application of legal incentives. In fact, the U.S. Congress has played a pivotal role in influencing innovation since the 18th century.

The history of fostering innovation in the United States is best illustrated by envisioning a pendulum sometimes swinging towards greater innovation, and sometimes swinging back. Given the ever-changing legal, economic and technological landscapes, stabilizing incentives for innovation can be a very difficult task, principally with copyright law. Therefore, as the following historical perspective demonstrates, innovation can be promoted, hindered and then promoted once again through the application of various legal frameworks.

---


16 Id.
The Constitutional & Theoretical Underpinnings of Innovation in Copyright Law

The envisioned pendulum first swung towards greater innovation when the United States’ Founding Fathers recognized the importance of providing incentives for artists and inventors to create. This recognition was primarily evidenced in the Patent and Copyright Clause of the U.S. Constitution which empowered Congress to “Promote the Progress of Science and the useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries.”17 As a result, the goal of today’s intellectual property law is to balance a free environment to create with monopolistic incentives allowing authors and inventors to have certain “rights” in their works.

The proclaimed intent of federal copyright law mirrors this constitutional goal and provides incentives for innovation and creativity by offering limited monopolies for original works of authorship fixed in tangible mediums of expression.18 These monopolistic privileges are codified in Title 17 of the United States Code as “rights” and include the following: the right to reproduce work; the right to prepare derivative works; the right to distribute copies of work to the public by sale or other transfer of ownership; the right to perform the work publicly; and the right to display the work publicly.19 Thus, a copyright is a very basic type of property interest that materializes automatically when an individual

17 U.S. CONST. art. I, §8, cl. 8.
reduces his creative idea to a tangible medium, such as the magnetic media of a hard drive, a CD-ROM, or software code.\textsuperscript{20}

The policy objective of U.S. copyright law is framed as a means, not an end, to the public’s benefit.\textsuperscript{21} Courts have emphasized this point by noting “the sole interest of the United States and the primary object in conferring the monopoly lie in the general benefits derived by the public from the labors of authors.”\textsuperscript{22} Thus, copyright protection is not conceived of as a natural right, but necessary because it stimulates progress and innovation.\textsuperscript{23} Fostering these public benefits however, requires balancing incentives with monopolistic outcomes: “without a legal monopoly not enough information will be produced but with the legal monopoly too little of the information will be used.”\textsuperscript{24} A look at the original intent of U.S. copyright law thus reveals how the government allowed the envisioned pendulum to swing towards innovation by offering authors unique benefits in return for their contributions to society.

\textsuperscript{20} See supra note 4, at 351.
\textsuperscript{22} Fox Film Corp. v. Doyal, 286 U.S. 123, 127 (1932).
\textsuperscript{24} ROBERT COOTER & THOMAS ULEN, LAW AND ECONOMICS 135 (1988), quoted in Paul Goldstein, Comments on a Manifesto Concerning the Legal Protection of Computer Programs, 94 COLUM. L. REV. 2573, 2574 (1994).
How Copyright Law Began to Fail Innovators and Hamper Technological Innovation

Like every property right, copyright “has an equal but opposite existence as a limit on inherent freedom.” The first U.S. copyright statute was the Copyright Act of 1790 and only afforded protection to maps, charts and books. The Copyright Act has been revised numerous times since then and its scope expanded to reach other mediums. It is this expansion (and accompanying copyright restrictions) that eventually forced the pendulum to swing back towards the side of fewer incentives for innovation.

FOSS proponents in particular have argued that the expansions in copyright law have unacceptably limited “inherent freedom.” Open source software supporters have noted for example,

with admitted overgeneralization, it is more or less fair to say that modern copyright law entitles copyright owners to prohibit others from doing almost anything with protected works, except to view, watch, or listen to the works privately, sell legally obtained copies (such as used books and CDs), and engage in “fair use” under the law.

The Copyright Act of 1976 and the Berne Convention Implementation Act of 1998 are two modern revisions of the copyright law that prompted such criticism. The Copyright Act of 1976 and the Berne Convention Implementation Act of 1998 together effectively ended the use of “formalities” and established that an author’s work was immediately protected upon reduction to a tangible

---

26 See Copyright Act of 1790, 1 Stat. 124 (1790).
27 See supra note 25, at 401-402.
28 Id. at 402 (emphasis added).
medium of expression—today there is no longer a need to register or provide notice that a work is copyrighted because protection is immediate.

According to supporters of increased copyright protection, the implementation of the Copyright Act of 1976 and subsequent expansions were necessary and reflected a utilitarian calculus that resulted in the public’s benefit. The math goes something like this: if the monopolies created by copyright law provide incentives to increase the production of “Science and the useful Arts,” then more protection equals more inventions and creations.

Modern scholars however, have aggressively come out in opposition of this utilitarian rationale and instead argue that “as the copyright balance tips in favor of greater protection, these expansions may cause inefficient outcomes,” resulting in greater burdens and more complex rules. This is the pendulum in action swinging towards less innovation.

One particular example showcasing how the expansions in copyright law burdened innovation played out in the United States Supreme Court case of *Eldred v. Ashcroft*, where a public citizen tried to create a digital library of public domain literature. The controversy arose over the inclusion into the library of a collection of Robert Frost poems which were scheduled to enter the public domain in 1998, but which were afforded greater protection (until 2019).

---

30 Goss, supra note 21, at 970.
31 Id.
32 Id. at 970-971.
34 A work falls into the “public domain” when “it is no longer under copyright protection or if it failed to meet the requirements for copyright protection. Works in the public domain may be used freely without the permission of the former copyright owner.” U.S. Copyright Office, Copyright Definitions, http://www.copyright.gov/help/faq/faq-definitions.html.
after the passage of the 1998 Sonny Bono Copyright Term Extension Act. The Court upheld the constitutionality of extending copyright protection to the disappointment of many authors advocating a less restrictive copyright. The fear was that copyright law was going too far by protecting commercial culture to the detriment of the public domain. As musicians, authors, and computer programmers noticed the expansive changes in copyright law they too sought a new legal framework where they could both protect their rights and encourage freedom in an innovative and collaborative environment. This was the beginning of the pendulum swinging back towards fostering innovation.

Traditional copyright law supporters might argue that expansions in the law are necessary to meet new technological demands, and that the pendulum never swung back in the first place. For example, the Digital Millennium Copyright Act of 1998 was an expansive copyright statute that, among other provisions, provided greater protections against Internet piracy – an important and growing concern. Along these lines, modern copyright supporters therefore argue that the law does a good job of fostering innovation mainly through the application of greater commercial incentives. Basically, the idea is that authors

---

36 See Goss, supra note 21, at 973-977.
37 See Forsythe & Kemp, supra note 18, at 346 (responding to the increasingly common perception that copyright is so broadly applied that it threatens the progress of science and the useful arts, rather than promoting it).
38 3 Melville B. Nimmer, Nimmer on Copyright §12.16[B] (1976) (“The digital environment poses a unique threat to the rights of copyright owners, and as such, necessitates protection against devices that undermine copyright interests. In contrast to the analog experience, digital technology enables pirates to reproduce and distribute perfect copies of works – at virtually no cost at all to the pirate. As technology advances, so must our laws.”).
and inventors will invent more when you offer them more money, even if there are more restrictions. But the next section offers one alternative view and showcases how the modern copyright framework may not be ideal in every situation; instead the law can be repositioned to appropriately balance incentives with freedoms.

How Software Programmers Altered the Copyright Landscape through the Application of FOSS

The real genius behind the FOSS movement was that participants utilized a uniquely tailored licensing scheme to reposition copyright law in order to promote collaboration and innovation in the software industry. FOSS participants expressly rejected the expansion of exclusive rights and instead aimed to create an environment where users could legally share work without giving up important rights. The progression of this movement illustrates how the envisioned pendulum swung once again towards promoting greater technological innovation.

Originally, the impact of copyright law on software programming practices was limited. Software source code became the subject matter of copyright in 1964, but the protection did not register on programmers’ radar until the 1970s when the expansion of the computer industry led to the creation of a separate software market. As software programs became more commoditized,

39 Shun-ling Chen, To Surpass or to Conform: What Are Public Licenses For? 2009 U. ILL. J.L. TECH & POL’Y 107, 112-13 (2009). See also Copyright Office Circular, No. 61 (1964), reprinted in 11 Bull. Copyright Soc’y 361 (1964) (This circular made software copyrightable in effect, but it wasn’t until 1980 when Congress passed the Computer Software Copyright Act that the definition of “computer program” was formally added into the federal copyright statute.”); Copyright Act of 1980, Pub. L. No. 96-517.
copyright became a powerful mechanism large firms could use to protect their ownership interests in carefully crafted computer programs.\textsuperscript{40}

An important consequence of the commoditization was that prepackaged software programs were routinely released without their source code.\textsuperscript{41} The rationale behind keeping source code secret was two-fold: 1) if users could see the source code then they would be able to extrapolate the steps the original programmer took in developing a highly advanced piece of code,\textsuperscript{42} and 2) hiding the source code protected the underlying copyright interests of the programs. The habit of keeping source code secret led to a variety of problems because programmers were no longer capable of collaborating and working-off one another in a mutually-beneficial environment. In addition, the secrecy fostered incompatibilities between various software and hardware components because programmers were encouraged to work individually.\textsuperscript{43} This expansive application of copyright law eventually prompted the development of FOSS.

The modern history of the FOSS movement began in 1969 with the creation of the powerful UNIX operating system.\textsuperscript{44} The movement gathered real steam in 1984 however when Richard Stallman proposed one of the most important open source projects in the history of the movement: a free UNIX-style

\begin{flushright}
\textsuperscript{41} Mann, supra note 40, at 8.
\textsuperscript{42} See supra note 4, at 357-58.
\textsuperscript{43} See Mann, supra note 40.
\textsuperscript{44} Id. at 11.
\end{flushright}
operating system called GNU. The goals, advanced by a collaborative team, were to 1) create a new operating system with the benefits of UNIX, and 2) keep the source code free. GNU became a viable operating system in 1994 when programmer Linus Torvalds contributed a working kernel, at which point the software evolved into GNU/Linux (or more commonly in recent years, Linux). A license was developed for this software called the GNU General Public License (GPL) and it allowed the public to use the source code as long as any modifications were reciprocally released under the same license.

In spearheading the Linux project, Stallman put his vision of free software into practice with a license permitting software releases without restrictions on copying and derivation. Theoretically the software could have been released into the public domain, but Stallman’s fear was that downstream users would just subsume the available source code into their own proprietary projects. Thus, through the open license, Stallman was able to prevent the “proprietization” of derivative software. Stallman’s actions reflect the movement’s guiding principles by providing “information that is truly available to its citizens – for

---

46 See Mann, *supra* note 40, at 11.
47 In computing, a “kernel” is the central component of an operating system. See generally supra note 23, at 161-162 (“In keeping with the collaborative roots of academic programmers and probably influenced by the efforts of Richard Stallman, Torvalds openly shared the source code of his kernel and eventually decided to license the project using the open source [GPL].”).
48 See Mann, *supra* note 40, at 11.
49 See Goss, *supra* note 21, at 964.
50 See *supra* note 45.
51 See *supra* note 15, at 368-69 (explaining the copyleft phenomenon as a form of “legal jujitsu,” where the author uses copyright to protect the free software by rearranging the restrictions downstream users will face).
example, programs that people can read, fix, adapt, and improve, not just operate.”52 This guiding principle is buttressed by the need to promote innovation. The idea is that the more people involved in a project, and the more freedom allowed, the more innovation that can result.

The software community began to recognize the validity of the FOSS movement’s activities on the heels of a number of open source success stories. For example, the Apache HTTP Server is an open source web server and the most popular on the Internet with greater than a 45% market share.53 A web server is essentially a computer program that manages web sites by handling requests to view and use the aforementioned web sites.54 Software industry experts analyzing the web server market report:

[E]vidence suggests that Apache has been at least as innovative as the others in introducing new features. Certainly there is no evidence here that the open source model was less able than the proprietary model to turn a basic experimental idea into a commercially viable product, or that it free-rides off of ideas developed in a proprietary market.55

Whether it is the Apache HTTP Server or other applications like the Mozilla and Google Chrome web browsers, most Internet users today rely on FOSS technology. Beyond providing benefits to the everyday Internet user, FOSS

54 BELLIA, BERMAN & POST, supra note 3. See generally Internet History, http://www.livinginternet.com/i/ii.htm (“Every web site is managed by a web server. The web server handles all of the network communications with individual user browsers. The server accepts HTTP requests for web pages, and sends the requested pages to browsers over the Internet ... Apache is now a sophisticated, multi-platform application, and the most popular web server in use on the Internet.”).
55 See supra note 15, at 333.
applications are increasingly being used in the private sector and have “become a fixture in tech-heavy disciplines such as financial services, military intelligence, online retailing, and next-generation cellular telephones.” Additionally, the FOSS model of collaboration and openness has spurred scientists in other fields to tackle challenges in new and innovative ways: “[o]pen source helped crack grand challenge science problems such as sequencing and the human genome.” Thus, FOSS participants have proven their ability to foster innovation in the marketplace.

The growth and success of open source software led to the formation of private organizations such as the Open Source Initiative, which today provides a cogent definition for open source projects: “open source doesn’t just mean access to the source code,” but requires that the distribution of FOSS comply with various criteria including that the license allow derived and modified works. As a historical note, the open source and free software movements started off in the same camp but in 1998 some members split off from the Open Source Initiative to form the Free Software Foundation. The two camps (“open source” software and “free” software) have some philosophical differences but ultimately both

---

56 Michael Tiemann, How Open Source Software Can Save the ICT Industry One Trillion Dollars Per Year, 4 (The Open Source Initiative, 2009), available at http://opensource.org/files/OSS-2010.pdf. Tiemann is the Vice President of Open Source Affairs at Red Hat and wrote this white paper for the Open Source Initiative.

57 Id.
strive to use open licensing models to promote innovation and collaboration.\textsuperscript{60} The paper thus refers to both groups as working within a singular movement.

The courts have also recognized the FOSS licensing scheme and correctly understood its position with regard to copyright law. In the case of Richard Stallman’s GPL for the Linux operating system, the Seventh Circuit noted in \textit{Daniel Wallace v. International Business Machines Corporation} that,

\begin{quote}
Copyright law, usually the basis of limiting reproduction in order to collect a fee, ensures that open-source software remains free: any attempt to sell a derivative work will violate the copyright laws, even if the improver has not accepted the GPL.\textsuperscript{61}
\end{quote}

When programmers first began utilizing FOSS licenses, the notion of open source code seemed counterintuitive to many people in the software community. Critics argued that traditional copyright laws were being subverted because allowing users to see source code meant they could replicate programs without paying for them.\textsuperscript{62} One theoretical precept of copyright law is that the availability of protectionist monopolies promotes innovation because author get leverage. This leverage, which restricts third parties from entering the space of the copyright holder, provides the author with an incentive to create. But with the FOSS movement, critics argued copyright law was being overturned.

The beautiful irony here is that while FOSS, and the open licensing scheme that inspires it, function according to a different set of rules the movement

\begin{footnotes}
\footnote{To clarify, the Free Software Foundation treats the approach to “free” software as an ethical inquiry; the organization promotes the availability of source code and the freedom to copy and modify because they are “rights.” The Open Source Initiative takes a more practical approach and “pitches free software because it works, not only because it’s the right thing to do.” \textit{Id.} at 268-70.}
\footnote{Daniel Wallace v Int’l Bus. Machines Corp., 467 F. 3d 1104 (7th Cir. 2006).}
\footnote{\textit{See supra} note 4, at 358.}
\end{footnotes}
still relies on copyright law. In fact, the FOSS movement is groundbreaking in that it is based on a repositioning of copyright law (not a rejection of it). The FOSS licensing regime is “copyright-based” because the authors use their afforded rights under the statute to grant conditional rights to others; “thus, at the bottom, open source software relies on the copyright status of computer programs as literary works, but deploys the rights arising from that status in a unique way.” Ultimately, the modern copyright ideology is flipped on its head (but not overturned) with FOSS facilitating greater freedom, instead of administering greater restrictions. In this way, the envisioned pendulum swings towards fostering greater innovation.

Furthermore, FOSS projects and licenses were deemed legally acceptable in the recently decided case of Jacobsen v. Katzer. The case before the Federal Circuit Court of Appeals dealt with the enforceability of one particular open license, the Artistic License, where a commercial developer refused to obey the obligations detailed in the license agreement. In upholding the validity of the license, the court ultimately implied that all open licenses would be enforceable under copyright law.

Nevertheless, no legal framework, governmental incentive, or judicial decision will likely ever be able to perpetually foster innovation because technological development and various economic cycles will continually

---

63 Id. at 357-58.
64 Vetter, supra note 10, at 71.
challenge society’s view of how innovation should be fostered. In economic
downturns policymakers may be more willing to regulate technology; while
robust economies might inspire policymakers to open up markets so more players
can take advantage of the opportunities. Therefore, the envisioned pendulum will
never remain stationary, and as the next section outlines, a new challenge
threatens to swing the pendulum back towards the side of greater barriers to
innovation.

**A Challenge on the FOSS Frontier: License Proliferation**

With the widespread adoption of FOSS principles and practices
throughout the software industry, a new problem has the potential to sidetrack the
FOSS movement’s progress towards fostering innovation. That problem is
license proliferation and it threatens to decrease incentives for innovation.
License proliferation refers to a slowly developing phenomenon in the FOSS
community where more and more open licenses are being drafted and adopted all
the time. The fear is not only that the number of open licenses is increasing, but
that new licenses are incompatible with previous FOSS licenses and difficult to
comprehend.

Some FOSS participants argue that license proliferation is actually a non-
issue because the majority of FOSS projects rely on only a few established open

---

70 See supra note 66.
licenses. But even mild license proliferation (often driven by new and inexperienced FOSS participants) can still be hazardous. The problem with license proliferation is perhaps best illustrated by the image of a house covered in climbing vines. At first, the vines enhance the character of the house in the same way that open licenses enhance the FOSS movement. As the vines grow and creep along the sides of the house, this too may not seem like a problem. But before the homeowner knows it, and without cutting away some of the foliage, the house is covered in a mess of vines completely overcoming the building. This image showcases why license proliferation is a problem: although it may not seem like the addition of new licenses threatens innovation, this is a “creeping” problem and if the licenses are not somehow maintained and manicured the movement may be overcome.

The Development of the “Creeping” License Proliferation Problem

There are a number of reasons explaining the recent expansion in open licenses including “author vanity”, license quality, and the need to meet new challenges in the ever-evolving software industry. The particularly harmful phenomenon of “author vanity” relates to the idea that license authors believe they need to create their own specific licenses to satisfy individual pet peeves; one scholar notes that author vanity “is a matter of lawyers (or programmers acting as their own lawyers) quibbling over esoteric legal issues that have little or no

---

71 Posting of Matt Asay to CNET News, The Open Road: The business and politics of open source, http://news.cnet.com/8301-13505_3-100007321-16.html (Aug. 5, 2008 9:07 ETD). Asay is an open source industry veteran having served as the Vice President for Business Development at Alfresco and he was one of the founding members of Novell’s Linux Business Office.

72 See supra note 67, at 277-78.
In addition, these so-called “vanity licenses” can also be generated by established software/Internet companies moving into the FOSS marketplace – these companies will appropriate a FOSS license, make minor adjustments, and then rename it with their trademark (or something similar) to maintain their brand name. Satisfying this type of vanity is no justification for the harms that result from too many varied open licenses. The Open Source Initiative lists over 60 open licenses available for use, yet in the Initiative’s License Proliferation Report only nine are widely used, with another 9 deemed redundant, and another 24 licenses “non-reusable.” The point is that the problem of license proliferation concerns not only the increasing number of open source licenses being created, but their variety and scope.

With license authors constantly making nuanced changes to license provisions, innovation is threatened in two ways: (1) the proliferation generates incompatible licenses, and (2) the proliferation encourages the use of extraneous legalese and complicated clauses which the average FOSS participant cannot understand. Both these problems are significant because they actively hamper innovation.

73 Id. at 278.
74 Posting by Mark Hinkle to Linux Today, http://blog.linuxtoday.com/blog/2008/05/the-curse-of-op.html (May 9, 2008, 09:35 EST) (discussing a situation where Sugar CRM, a company which provides open source customer relationship management applications, wanted to certify their own FOSS license, the Sugar Public License. The license was a modified version of the already approved and widely-used Mozilla Public License.)
75 See supra note 66.
76 See supra note 67, at 263.
Problem 1: Incompatibility

Lawrence Rosen, who served as general counsel for the Open Source Initiative and has authored open licenses, uses the analogy of a Mozart symphony in describing the role of incompatibility in FOSS license proliferation,

As a license musician, I’m not bothered as much by “too many notes” as I am by the fact that the notes aren’t always in the same key. License proliferation has become an important problem because software under those different licenses cannot always be played consistently and compatibly everywhere.\(^{77}\)

When Rosen analogizes and mentions that he is not bothered by the “many notes” in a Mozart symphony he underscores that the licenses themselves, while increasing in number, are not a problem. Rather, it is the application of these licenses in incompatible ways that is problematic.

One scenario dealing with incompatible licenses occurs when a programmer is encumbered by what can be done with licensed code under a subsequent license. Take for example a situation where content under one open license does not require that the user provide attribution, while content from a second license requires both attribution and that modifications be made public; in this scenario there is a limit on what content can be used under the first license.\(^{78}\)

The programmer can continue to face such unworkable dilemmas when additional FOSS source code is used that is more prohibitive.\(^{79}\) Innovation is derailed in such situations because the software programmer is forced out of a collaborative environment since the incompatible licenses have limited his options.

\(^{77}\) Lawrence Rosen, License Proliferation, 1 (2005), http://www.rosenlaw.com/LicenseProliferation.pdf.


\(^{79}\) See supra note 67, at 282-83.
License incompatibility can also occur when a programmer is faced with two licenses that seem very similar (because they both require modified content be released freely), but are in fact irreconcilable because each requires that modified content be released under their individual licenses. Take for example a situation where License A and License B are both FOSS licenses that require source code be released under their respective licenses. What if a software programmer wants to use source code under License A and License B in his new program? How does the programmer license the new program if the code has to be released both under License A and License B? The noted Internet law and open source scholar Lawrence Lessig writes that this form of incompatibility is a “product of bad design” and particularly dangerous because it results in content being “ghettoized into particular licenses, fracturing the resources for innovation.”

These incompatibility outcomes discourage the FOSS movement from moving forward because programmers are either denied the ability to use code altogether or are unable to utilize various pieces of source code in one project. The restrictions decrease the power of the FOSS movements’ ability to attract participants. Without the participants working in a collaborative environment and taking advantage of the groundbreaking open licensing regime, innovation is threatened.

---

80 See supra note 76.
**Problem 2: Lack of Comprehension**

License proliferation also hampers innovation when it yields unintelligible license provisions too complicated for the average programmer to understand.\(^8\) While many FOSS programmers are actually well-versed in the underlying principles of open licenses, the problem lies in the increasing number of nuanced open licenses. The programmer now faces the challenge of understanding the exact rights and obligations specified in scores of licenses. The lack of comprehension can be particularly threatening to innovation because programmers may be dissuaded from participating in the FOSS movement when one misstep may lead to a claim of copyright infringement, injunctions and costly lawsuits.\(^8\)

For example, *Jacobsen v. Katzer* stands for the proposition that anytime a third party steps outside the bounds of a FOSS license agreement, they will be considered infringers.\(^8\) If programmers do not understand the parameters of any one license agreement and either intentionally or mistakenly break the terms of the agreement, they may be subject to a lawsuit. In *Jacobsen* the plaintiff created a FOSS software application that enabled users to control model trains through their personal computers.\(^8\) The terms of the license allowed anyone to download the application for free, but if a programmer wanted to redistribute a modified version they would have to provide attribution for the plaintiff and notification

---

\(^8\) See Forsythe & Kemp, *supra* note 18, at 362. (“A disadvantage is that licenses need to be easy to understand, since most users are not trained in the law and its particular use of language.”).

\(^8\) *Supra* note 67, at 280.

\(^8\) *Jacobsen v. Katzer*, 535 F. 3d. 1373, 1382-83 (Fed. Cir. 2008).

\(^8\) *Id.* at 1375.
that the program had been altered.  The defendant in the case assimilated the plaintiff’s code in his own commercial application for model train hobbyists without providing attribution or notification.  The Federal Circuit Court of Appeals held the license enforceable and implied that anytime a third party acts outside the scope of a FOSS license agreement, even if they don’t understand the terms, they may be liable for copyright infringement. Thus, without an exact understanding of what license terms mean (for example, the difference between attribution and notification), FOSS participants take great risks in using available source code in their innovative endeavors. These risks may discourage programmers from participating in the movement.

Again, the power of the FOSS movement lies in its open licensing regime geared towards fostering innovation. Without end users taking advantage of the FOSS licenses and contributing to the communal body of source code, the development of new and inventive programs is seriously threatened. In order to further innovation, the solution to license proliferation must therefore take into account both the problems of incompatibility and the highly nuanced, yet potentially unintelligible licenses.

\[85\] See supra note 65, at 314-315.  
\[86\] See Jacobsen, 535 F. 3d. at 1379.  
\[87\] Id. at 1382-83.  
\[88\] Attribution refers to the idea that a programmer who modifies and distributes FOSS source code, must credit the original developer of the code. Notification refers to the idea that a programmer who modifies and distributes FOSS source code, must notify downstream users that they modified the code. These are basic concepts and both attribution and notification provisions may vary greatly among licenses, particularly with regard to who and what must be credited and notified.
Accounting for the Other Side and Why New Licenses May Be Necessary

Not all new licenses are harmful. The software industry is dynamic and technology will keep evolving. Therefore, the FOSS licenses that will support technological innovation will have to be flexible. The fear of license proliferation should not mean that all current licenses be invalidated. Rosen voices this notion by arguing that if a symphony has too many notes, the solution is not to throw them out.\(^8^9\)

FOSS licenses should be introduced and adopted but only when new industry practices and technological advances call for revisions and adaptations. In order to facilitate the introduction of new licenses, FOSS participants will have to balance the need for change with some of the harmful results of license proliferation, namely incompatibility and lack of comprehension.

Ending License Proliferation and Getting Back to the Mission at Hand: Fostering Innovation

Why Instituting a Moratorium on the Creation of New Licenses is Neither Practical or Recommended

At first, the idea of instituting a moratorium on the creation of new licenses may seem comforting. A moratorium would put a stop to the creation of new licenses until a solution to license proliferation was agreed upon by the FOSS community.

The Open Source Initiative and Free Software Foundation are major players in the FOSS community with enough influence to change licensing behaviors. In fact, one of the functions of the Open Source Initiative is to approve

\(^8^9\) See supra note 75.
licenses that meet the “Open Source Initiative Definition” and thereby ensure that authors are drafting licenses that subscribe to the tenets of the FOSS movement.\textsuperscript{90} The Free Software Foundation maintains their own definition of what “free software” is and the organization developed their own widely used license, the GPL.\textsuperscript{91} Both these organizations play a major role in deciding what licenses will be available for the FOSS community. Thus it makes sense that they should play a part in ending the harmful effects of license proliferation. With the moratorium, no new licenses would be approved by the Open Source Initiative until a workable solution was put into practice.

Nevertheless, the institution of a moratorium is not practical. There are times, possibly in the near future, when new open licenses will be required to meet unforeseen needs. The institution of a moratorium does not directly resolve the issue of license proliferation and promote innovation. The FOSS community needs a concrete solution that all participants, from programmers to legal practitioners, can apply.

\textit{The Real Solution to License Proliferation Lies in the Application of an Analytical Framework that Predicts When Innovation is Fostered}

This paper proposes that the problems associated with license proliferation can be solved if programmers and practitioners are provided with an analytical framework they can apply to specific license provisions. The goal of the framework proposed in this paper is to aid programmers and legal practitioners in understanding what individual open license provisions do so that they can 1)\textsuperscript{90}

\textsuperscript{90} The Open Source Initiative, Open Source Licenses, http://www.opensource.org/licenses.
\textsuperscript{91} The Free Software Foundation, GNU General Public License, http://www.fsf.org/licensing/licenses/gpl.html.
decide which existing licenses are beneficial; 2) discard any unnecessary licenses being used; and, 3) decide if a new license is necessary. This framework should be easy to understand so that any programmer or legal practitioner can apply it when considering the adoption of a new open license: if the new license enhances and promotes innovation (along the lines set forth below) then it should be allowed to enter the FOSS domain.

Given that the FOSS community prides itself on fostering innovation in a collaborative environment, it will be crucial that all parties understand how certain license provisions provide incentives to innovate. The application of established social science theories on innovation can be helpful in this endeavor. With the advent of the FOSS movement, social scientists began studying the community in order to understand how programmers were generating technological innovation without the incentives provided by traditional copyright law. Among the various studies, social scientists focused on the theory of “open innovation” in explaining the ability of the FOSS community to generate technological innovation. The “open innovation” theory suggests that collaborative and decentralized environments have the ability to generate more

---

92 One important notion resulting from recent economic studies on innovation and FOSS is that there is no reason to believe that monopolistic protections are necessary to promote innovation. See supra note 15, at 327 (Arguing that three economic studies considering the economic implications of open source software by Hann, Roberts, Slaughter, and Fielding; Lerner and Tirole; and Llanes find that innovation can in fact thrive in an open source environment).

93 See also Joel West, Open Source and Proprietary Models of Innovation: Beyond Ideology: PART I: Business, Law, and Engineering Perspectives on Open Source Innovation: Policy Challenges of Open, Cumulative, and User Innovation, 30 WASH. U. J.L. & POL'Y 17, 22 (2009). (“Like cumulative and user innovation, open innovation builds upon the assumption of dispersed capabilities for identifying and implementing innovation. Although user innovation and cumulative innovation focus on consumer welfare, the open innovation paradigm emphasizes the opportunities for profit and competitive advantage by individual firms.”).
robust technologies than the confines of a centralized network, such as a single firm or laboratory.\textsuperscript{94} This paper adopts the social science theories of open innovation to analyze license provisions. In particular, the principles of the “collaborative web” theory of innovation,\textsuperscript{95} a sub-theory within “open innovation,” are applied to analyze innovation in FOSS licenses for two reasons: 1) because the complex, collaborative system described in the theory fits the FOSS model so well,\textsuperscript{96} and 2) because the “collaborative web” theory of innovation breaks down various properties of innovative communities and this categorization is helpful in dissecting license provisions at the micro-level.

For the FOSS community, the creation of deeply tied networks is natural. In collaborative webs, there is frequent communication between members and a multitude of discoveries in a complex, decentralized network.\textsuperscript{97} The parameters of such a collaborative network describe exactly how the FOSS community works. Thus, the collaborative web theory does not need to be modified to analyze the FOSS movement.

Beyond the parallels between collaborative webs and the FOSS community, is the methodology behind the sociological theory that is beneficial to the study of innovation in FOSS license provisions. According to researchers in


\textsuperscript{96} Id. at 315.

\textsuperscript{97} Id. at 313.
this field, innovation is more likely to occur in collaborative webs when the following properties are present:

**Connectivity**: The parts of the system are all connected to one another.

**Communication**: The parts constantly communicate with one another, sending rich and complex information.

**Self Organization**: The system re-organizes itself in response to environmental changes without central control.

**Constant change and flow**: Complex systems are never static. Even when they seem not to be changing, that statis is in fact maintained by constant activity.

**Disruptive innovation**: Even in an apparently stable system, if you know exactly the right place to act, you can often cause a sudden, dramatic change. [Disruptive innovation, framed by author Clayton Christensen, is where products or systems create new markets.]

**Heavy parallel**: Inspiration, selection, and development are all occurring all the time and are distributed throughout the system.

**Constant failure**: Many individual inspirations never get selected; many selections never get developed; many developed ideas never emerge from the system. This is not a problem to be corrected; in fact, it is a sign that this is a truly creative system.98

Researchers have reported that the FOSS community thrives precisely because it promotes all seven categories of the collaborative web.99 FOSS programmers contribute substantial time and effort to work within a system that is deeply connected, communicates often, is generally self-organized, constantly changes, witnesses disruptive innovation, is heavily paralleled, and is not afraid of failure.

The remainder of this paper will scrutinize popular FOSS license provisions and analyze each one with regard to the seven categories of the

98 *Id.* at 316-17.
99 *Id.* at 321.
“collaborative web” theory of innovation. The ultimate goal of this analysis is to understand how license provisions independently work to foster innovation. This is the type of analysis that programmers and legal practitioners will need to undertake in order to promote innovation in the face of license proliferation.

The Seven Category Analytical Framework in Action

Analyzing common FOSS provisions in a detailed fashion is important because it allows the community to look at what the provisions are actually doing in the system: are they increasing communication? Are they prohibiting constant flow and change? With this type of framework in hand, programmers can balance their need for a unique license with a licenses’ effect on innovation.

a) Derivative Works Right Provision: The *sine qua non* of FOSS licenses applies to the derivative works right of copyright.\(^{100}\) The derivative works right was originally conceived of in copyright law and ensures that the owner of a protected work has a proprietary interest over subsequent modifications and variations of that work.\(^{101}\) It means, for example, that Sylvester Stallone, who wrote the script and played the character of “Rocky Balboa,” has copyright protection over and above any third party who might write their own screenplay with a plot permutation using the same “Rocky Balboa” character.\(^{102}\)

---

\(^{100}\) *See supra* note 19.

\(^{101}\) 17 U.S.C. §101 (2009) (“A ‘derivative work’ is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgement, condensation, or any other form in which a work may be recast, transformed, or adapted.”).

\(^{102}\) Anderson v. Stallone, 11 U.S.P.Q. 2d 1161 (C.D. Cal. 1989) (This is a landmark case in copyright law explaining the implications of the exclusive derivative works in traditional copyright law).
In the case of FOSS licenses however, the derivative works right is granted broadly to downstream users so that they may modify, improve, and create new works based on the available source code. In the FOSS world, a computer programmer would license the code he developed, say the “Rocky Balboa Code,” and any downstream user would be allowed to use the “Rocky Balboa Code” as long as they ascribed to the provisions in the open license. Most open source projects, including Apache and Linux, operate in a collaborative environment and rely on the right to modify others’ work. These projects allow downstream programmers to modify and debug the software with the idea that these third parties will have better information, including whether a problem is due to the software or a user’s individual computer.103

The derivative works right is so important to the FOSS movement that it is in fact a necessary component in the Open Source Initiative’s definition of an open source license: “The license must allow modifications and derived works.”104 The reason for the heavy emphasis on the right to prepare derivative works is that “software has little use if you can’t maintain it (fix bugs, port to new systems, make improvements), and modification is necessary for maintenance.”105 But more than the practical need for a broad derivative right is the ability of the provision to substantially further innovation.

103 Vetter, supra note 10, at 80.
104 Supra note 56.
The derivative works right allows FOSS development to flow free-form in a decentralized and non-hierarchal structure, and therefore promotes constant flow, self-organization and disruptive innovation. Comparing and contrasting proprietary and open source models here is helpful: Microsoft, with its tight grasp on traditional copyright, releases complete overhauls of its operating system every few years, whereas the deployment of the open source Linux operating system is incremental and frequent under the Fedora Project. The Fedora Project is completely powered by the FOSS ethos and permits anyone to download the basic FOSS operating system for free and to make contributions to the code. Thus, if a third party programmer finds a bug in the system they can become a contributor and write the code to fix it. The Fedora Project is now on its 12th version and prominently displays on their website, in bold, that it is “free for anyone to use, modify, and distribute.” As a result of this freedom, more features and value-adds are released more frequently to the general public, more individuals participate in modifying the code, and this constant outpouring of support serves to increase innovation. In fact one recent Fedora Project development is PakageKit, a program which makes it easier to install and upload software on a computer, and in the words of the creator: “the primary design goal is to unify all the software graphical tools used in different distributions … to make the process suck less.” The open source nature of the Fedora Project has resulted in dozens


107 Id.

108 The Fedora Project, Homepage, http://fedoraproject.org/

109 PackageKit, What is PackageKit, http://packagekit.org/pk-intro.html. A list of the contributors to PackageKit can also be found at http://packagekit.org/pk-authors.html.
of new technologies aiding in the easy use and reliability of the Linux operating
system.\textsuperscript{110}

The idea that end users know that they are allowed to manipulate
particular pieces of source code without repercussion (as long as they abide by the
other provisions of an open license) allows the whole network to be fluid and
flexible. Social scientists and economists studying the motivations of software
programmers who joined the FOSS community found that about half of the
programmers “joined because they wanted to share knowledge and skills and this
number increased to 67\% when asked their reason for staying.”\textsuperscript{111} The
collaboration and fluidity promoted by the derivative works right means that in
fact “inspiration, selection, and development are all occurring all the time and are
distributed throughout the system.”\textsuperscript{112}

The derivative works right provision also furthers innovation with regard
to communication, connectivity, and heavy parallel. The derivative works right
enhances connectivity by fostering a network that connects different users based
on their mutual desire to work on collaborative FOSS projects. Software
programmers often rely on “libraries” of code to develop customized programs
and they generally prefer not to “reinvent[] the wheel.”\textsuperscript{113} The derivative works

\textsuperscript{112} See supra note 93, at 315.
right in FOSS allows programmers to connect and share code in this way without
the fear of a traditional copyright infringement lawsuit.

The derivative works right, however, does not generally promote constant
failure. One of the major goals of the derivative works right in open licenses is to
allow programmers to fix software problems quickly by allowing a large number
of users to work on various projects at the same time. This discourages the
furtherance of the constant failure category because programmers quickly work to
fix problems. Ultimately however, six of the seven categories are promoted in the
innovation framework and this analysis demonstrates the ability of the derivative
works provision to promote innovation.

b) Reciprocal Obligation in Distribution Provision: This provision
generally requires that a derivative work must be distributed under the same
license that was applied to the original open source code. This provision is
notably embodied in §5 of the General Public License Version 3 (GPLv3) which
institutes a reciprocal obligation by requiring that if a user wants to distribute
modified work of any GPLv3 code they “must license the entire work, as a whole,
under the [GPLv3] to anyone who comes into possession of a copy.” Upon a
cursory examination, the provision both fosters and slows innovation. Therefore
a detailed, analytical framework can provide a lot of information about how
exactly the provision functions in the face of innovation.

---

114 Supra note 56.
115 Supra note 89. See also Press Release, Free Software Foundation, FSF releases the GNU
General Public License Version 3 (June 29, 2007), available at http://www.fsf.org/news/gplv3_launched ) (At the time, the FSF offered this brief historical overview: “The GNU GPL is the most widely used free software license worldwide: almost three quarters of all free software packages are distributed under this license. It is not, however, the only free software license.”)
The reciprocal obligation provision promotes connectivity. The obligation that attaches to the license will touch every user that decides to take advantage of the code, thereby connecting different parts of the network. The parameters of how and if a reciprocal obligation should be applied downstream varies by license. Most licensors simply wish to ensure the integrity of the code, and that later users will not add further restrictions to the original source code.116 Thus, the reciprocal obligation provision sends a clear message to downstream users as it aids communication.

The ties created by the provision are very forceful however, and as a result may dissuade the constant flow of information. The reciprocal obligation provision in the GPL, also referred to as the “copyleft” provision, is known as one of the most restrictive license provisions.117 More specifically, the provision is critically regarded as “viral” because it forcefully attaches to a piece of code and may affect whether downstream users decide to take advantage of the original work.118 The “virus” attaches if any part of the program’s source code is covered by the GPL, and thus even if the programmer planned to release the code in a proprietary application, it would now have to be completely “open source” – hence, the infectious nature of the license provision.119 The forcefulness of the obligation limits communication by setting parameters around the subject matter. The limit on communication accordingly may dissuade programmers and thereby

---

116 Supra note 56.
117 Supra note 89.
118 See supra note 23, at 182.
119 See supra note 111, at 143.
slow innovation. Consequently, a seven category analysis begins to illustrate the various ways in which a license provision may act on innovation.

Taking this analysis further, the reciprocal obligation provision goes a long way to aiding self-organization by prescribing forceful norms for the community. The FOSS community consists of a complex network of programmers and end users whose actions are dictated by the “metes and bounds” of the community’s rules, specifically as set forth in the open licenses. License provisions which foster this connectivity and contribute to the development of strong norms, without central control, can lead to sophisticated and robust innovations. In this way, the reciprocal obligation provision promotes self-organization which in turn, fosters innovation.

Applying a seven category framework to the reciprocal obligation in distribution provision shows both how innovation is furthered and impeded. This license provision is in fact quite controversial in the FOSS community. Because the license provision has the potential to reduce the amount of incentives available for fostering innovation, a software programmer or practitioner would have to be very, very careful about including it in their FOSS license.

c) Attribution Provision: Many popular open licenses contain a provision which dictates that any subsequent release of a modified piece of code credit the original programmer or FOSS project. This attribution provision, and open

---

120 See supra note 23, at 189.
121 Id. at 161-162.
122 Attribution may include, but is not limited to, the following: recognizing the original author, the project which released the work, and contact information for the original author. An example: “You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of
licenses which allow almost any action with the caveat they receive attribution, are considered some of the least restrictive. 123

Attribution in the FOSS community is particularly important (although not always required or desired) because it helps a programmer gain status in the community. Open source software pioneer Eric Raymond has noted that requiring attribution for an author’s particular piece of code is critical given the fact that programmers often contribute many hours of free time to the development of a project simply because they value the admirable reputation they may receive as a result. 124 Accordingly, the Attribution Assurance License, a certified Open Source Initiative license, contains a provision requiring the author be named in the “hopes that its promotional value may help justify the thousands of dollars in otherwise billable time invested in writing this and other freely available, open-source software.” 125

When analyzing the attribution provision along the seven categories of the “collaborative web” theory of innovation, the attribution provision heavily promotes connectivity. Via the attribution provision members of the community know exactly how they are connected to one another and the system as a whole becomes more interconnected. 126 Attribution is an effective mechanism for connecting users because it encourages programmers to participate: “by properly

---

123 See Vetter, supra note 10, at 74.
124 See supra note 7, at 187-88.
126 See supra note 93, at 316.
attributing an author’s work as a matter of law ... many authors who might otherwise not contribute to open source efforts could be enticed to do so because they have less reason to think that their work will go unnoticed.”127 In this way the attribution provision promotes connectivity and ultimately helps foster innovation.

The culture of programming revolves around using other developers’ “tried-and-true” code and the attribution provision ensures that the programmer knows where and whom to seek if they want to use a particular piece of developed code128 – again supporting the connectivity category. The Apache License, 2.0 released in 2004 states that any distributed work must contain attribution for the original author,129 and allows the downstream user to require any other attribution notification they deem appropriate.130 Such attribution provisions go a long way to connecting a complex and elaborate community. In Apache’s case, any downstream user, no matter how far removed, will be aware that a particular piece of code was originally developed under Apache.131 This type of permanent attribution creates solid ties between members of the community. Along these lines, the attribution provision also encourages communication and the constant flow of information because as users recognize

127 See supra note 104, at 1769.
128 Id. at 1766-1767.
129 Apache Software Foundation, Apache License, 2.0, available at http://www.apache.org/licenses/LICENSE-2.0 (“You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works.”).
130 Id. (“You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.”).
131 See supra note 104, at 1782-83.
who else they are connecting with, it becomes easier to establish ties, communicate, and exchange information and ideas.

On the other hand, the attribution provision does not seem to promote self-organization one way or another because receiving credit for developed source code says nothing about a particular person’s authority. Attribution merely serves to identify the players, not rank them or set up a decision-making structure.

The attribution provision discourages heavy parallel and constant failure from happening by connecting users and theoretically preventing the same actions/mistakes from reoccurring. Given the collaborative nature of the software industry most downstream users are generally going to attempt to find a piece of developed code before trying to write code on their own\(^{132}\) – thus attribution promotes more ingrained networks that cut down on heavy parallel and constant failure.

Both programmers and license authors have the knowledge to partake in this type of analysis. By going through the framework category by category, FOSS participants will determine how license provisions are functioning with regard to innovation. This type of analysis, by all parties within the system, is the best way to ensure that the FOSS license domain is appropriately manicured.

**d) Notification Provision:** Some licenses require notification to the public about the changes the user made to the software.\(^{133}\) This provision goes beyond mere attribution for the original author or satisfying some reciprocal obligation,
but instead ensures that if the downstream user’s program is faulty (whether purposely or not) the original author will have separated himself from the damaged work.\footnote{See supra note 103 (“A concern among some software authors is that [the derivative works right provision] could allow unscrupulous people to modify their software in ways that would embarrass the original author. They fear someone deliberately making the software perform incorrectly in a way that would make it look as if the author was a poor programmer. Others are concerned that software could be modified for criminal use, by the addition of Trojan horse functions or locally-banned technologies such as cryptography.”).} The provision attaches to code when a derived work is modified and then distributed to the public. For example, imagine a situation where a financial expert moonlighting as a software programmer wanted to make some improvements to iFreeBudget, an open source application for personal finance management.\footnote{Id.} iFreeBudget is licensed under the Apache License 2.0,\footnote{Id.} which contains a notification provision, and therefore the financial expert would have to include notice of any work he contributed when releasing a newer version of iFreeBudget.

The notification provision is used in FOSS licenses because programmers fear being associated with bad or unwanted code. The fear ranges from being associated with a piece of source code that deliberately infects other programs via a cyber-virus to the benign concerns of a software development firm wanting to separate different versions of a product. For example, only Netscape can use the name Netscape Navigator for their web browser, but they have also released free FOSS versions of the web browser which other programmers can modify, but only under the Mozilla name.\footnote{See supra note 103.}
The notification provision is particularly useful for increasing communication in the software community because it apprises downstream users that a particular piece of code has been modified and thus helps to send “rich and complex information” across the network.\(^{138}\) Especially when benign, such notification helps connect the software community and “this networking may lead to true collaboration between individuals, in turn, to the creation of valuable creative works that would not otherwise have existed.”\(^{139}\)

While connections are created within the system, notification provisions may also separate different players at the same time. This effort to separate projects (whether “good” from “bad” or “mine” from “yours”), impedes constant failure and heavy parallel because when a piece of bad code is found, users can go back to the original, workable pieces and do not have to replicate failure.

Again, this type of analysis allows FOSS participants to see what a specific license provision is doing with regard to innovation. In the case of the notification provision, the application of the framework reveals that the provision fosters innovation by furthering connectivity and communication, but slightly slows innovation by hindering heavy parallel and constant failure.

e) Special Privileges Provisions: These provisions deal with situations where license authors feel the need to add a caveat to their licenses (while still remaining true to basic FOSS principles).

An example of the special privileges provision in action concerns the Netscape Public License which provided special privileges solely to Netscape.

\(^{138}\) See supra note 93, at 316.
\(^{139}\) See supra note 25, at 420.
Basically, Netscape reserved the right to re-license modifications any unrelated programmer made to their code; this included the right to make those modifications private, improving them, or refusing to give the result to the programmer.\(^\text{140}\) According to Netscape, the provisions were necessary because when the company moved from a proprietary software business model to an open source model they had to honor certain third-party contracts.\(^\text{141}\) But, this type of provision can be particularly detrimental to innovation: not only does it impede connectivity by fragmenting the system but it prohibits self-organization with one party dictating restrictive terms to others; breaks down the level of communication; and, does not allow for the constant flow of information.

Any special privileges provision has the potential to menace other members in the FOSS community, and as a result the provisions tend to elicit a lot of controversy and criticism. In Netscape’s case, for instance, the company ultimately decided to develop yet another open source license, the Mozilla Public License, which did not include the special privileges provision.\(^\text{142}\) The application of the seven category framework foreshadows this type of controversy by revealing the ways in which a special privileges provision may slow innovation. When programmers and license authors apply the framework before deciding on a special privileges provision they will be able to predict how the license provision will function and likely understand how the provision effects innovation.

\(^{140}\) See supra note 103.

\(^{141}\) Id.

\(^{142}\) Id.
Summarizing the Seven Category Framework’s Role in Ending the Harmful Effects of FOSS License Proliferation

What does the seven category framework mean for innovation?

Ultimately, it is a tool programmers and license authors can use to understand FOSS licenses and their role in fostering innovation. When a programmer looks at all the licenses in the FOSS domain they will be able to use the framework, with its seven categories, to analyze the provisions of a particular open license. By systematically analyzing a license, provision by provision, the programmer will gain a deeper understanding of what the license is doing. This understanding will in turn allow the programmer to both decide if a license is appropriate for his purposes, as well as avoid licensing litigation.

The analysis undertaken in the section described above should act as a model for how each programmer and practitioner should apply the seven category framework. While this individualized analysis might result in different outcomes for different programmers and practitioners, it will be important for innovation to remain a frontline motivator in the FOSS community. License proliferation has threatened the promotion of innovation. The application of a framework supremely centered on the promotion of innovation is a forceful way of ensuring that FOSS participants stay aligned and in tune with the idea that the law can keep up with technology.

Conclusion

… from 1996-2006, open source software consistently doubled in size (and presumably capability) every 12.5 months, to 1 [b]illion source lines of code (SLOC). This is astonishing both because using conventional metrics this would have an implied cost to produce of more than $25 [billion U.S. dollars], and because such a
rate of production implies that open source software has achieved what proprietary, notoriously has not: a Moore’s Law for software. And this has enormous implications, because exponentially lower production costs means faster and greater return on investment and a lowering of barriers to innovation...\(^{143}\)

FOSS is not going anywhere – the number of open source projects continues to increase as the public recognizes the new and improved technologies that can result. Some of these FOSS projects include the widely-used Mozilla web browser and programs such as Pidgin, which allow users to operate all their Instant Messaging accounts at once.\(^{144}\) OpenOffice.org is an advanced open source office and word processor suite that is compatible with all other major suites, is free to download, use and distribute.\(^{145}\) By taking copyright law into their own hands, FOSS participants have refashioned the software framework. The results of this endeavor have distributed innovative technologies to all end users. Now, the community must once again come together to combat the problem of license proliferation.

With license proliferation, the fear is that the FOSS movement may be going too far and straying for programmers’ original intent to create an innovative, creative and collaborative environment. The ability of FOSS participants to understand open license provisions and use them to further innovation is paramount. In administering the FOSS licensing regime, programmers and legal practitioners must adopt an analytical approach to understanding license provisions. The framework provided in this paper should

\(^{143}\) *Supra* note 56, at 2.

\(^{144}\) *See* Lifehacker’s Best Open Source Software, *available at* http://lifehacker.biz/articles/best-open-source-software/, *for a listing of top FOSS programs*, including Firefox and Pidgin.

\(^{145}\) *Id.*
be used as a tool to steer clear of the harmful effects of license proliferation. By applying this framework, FOSS programmers and license authors will be able to ensure that the goal of FOSS, the promotion of innovation, remains intact.

It is true that the law can be outdated and slow to adapt to societal, environmental and technological changes. Generally the law catches up to these changes in three ways: 1) lawmakers are put on notice and appropriately tailor the law to suit society’s new needs; 2) the judiciary correctly understands the direction of these changes and appropriately applies the law; and, 3) legal practitioners act within the bounds of the law to meet new societal, environmental and technological demands. This paper addresses the third category by focusing on the way software programmers write and use licenses and how practitioners conduct license analysis. FOSS participants have already proved once their ability to reposition the law in a way that fosters the original intent of copyright law. They have shown that the law can keep up with advances in technology and the benefits of technology can flow to all. Now it is up to the FOSS community to continue fostering innovation by ending the license proliferation problem.