Imagining and Reimagining Open Software Patents

Micah C Thomas
Imagining and Reimagining Open Software Patents

By

Micah Chaim Thomas
# Table of Contents

I. Introduction ......................................................................................................................1

II. Historical Overview ........................................................................................................5

II. Costs and Benefits of Open Software Patents ...............................................................11
   A. Philosophical Considerations ....................................................................................11
   B. Social and Economic Costs and Benefits .................................................................14
   C. Legal Considerations for an Open Patent .................................................................18

III. ‘Open’ ..........................................................................................................................26
   A. Open Licenses Touching Patents ............................................................................28
   B. Pledges ....................................................................................................................34

IV. Conclusion ...................................................................................................................41
I. Introduction

Software is being patented and if the open source community can find benefit in holding patents, then the body of experience developed from using copyright should be updated to include patent specific licensing and practice standards. The legal protection of software has run the gamut of intellectual property. The bounds of that protection are naturally limited by the category of IP being used, as well as by the transaction model employed for distribution. As the number of software patents grows, the social, legal and philosophical implications upon competing models are becoming real issues for the entire industry. This paper is directed at the open source community. Although the voices in the community are divergent, sometime conflicting, as a whole FOSS is still largely ideologically rooted in common principles. The community has grown accustomed to the utilization of copyright in combination with creative licensing. Software patents have been relegated to troublesome and dangerous nuisance by the community as evidenced in FOSS license patent provisions. The primary concern is that patents will be enforced against the community, but the spill over effect is that a negative perception of patents has forestalled any desire to acquire, prosecute, and enforce patents for FOSS itself.

Patented software permissions employing an open license must have differences from those permissions under copyright. Those differences relating to the different exclusive rights between the two types of protection should receive attention when drafting a license, because the enforceability of those licenses depends on it.
Notwithstanding drafting challenges, an open patent itself may be useful to uphold the principles of the open source community. If the community fails to self-define the contours of an open patent, outsiders representing major non-open source principles will, as they have already begun to, appropriate the term and craft a competing definition.

Throughout this paper the terms open, open source, open source community, all reach towards a description of the social movement whose core principles are: access to source code, and maximum freedom of usage, development and distribution. With programs generally known as free and open source software, FOSS stands for the accepted and known value of the larger community.¹

The foil to FOSS suffers from much of the same linguistic wrangling common to the growing pains of a neologism. The usual terms, each inarticulately failing to distinguish the software as a class from FOSS include, “commercial software, proprietary software, and closed source software.”² This paper uses the catchall label of Binary Use Software (BUS) because of the apolitical focus on the meaningful licensing differences from FOSS. BUS, in most general transactions, differs from FOSS in the “access to the

¹ Ronald J. Mann, Commercializing Open Source Software: Do Property Rights Still Matter? 20 Harv.J.L&Tech. 1, 3 (2006). Even in law review articles, there seems to be a continuing misunderstanding about the word ‘open.’ Mann has a very insightful article, but when writing about whether or not open source projects can coexist with software patents he writes, “the major open source projects must acquire patents of their own or they must rely on the patent portfolios held by those who participate in the proprietary model.” The problem with this statement, echoed across the web and in academic writing, is that open source copyrighted software is proprietary. The mere fact that FOSS gives, or more accurately licenses, away some of the rights does not make their authority to do so any less proprietary.

software’s source code and the nature of a licensee’s right to do things with the software.”\(^3\) BUS also arises out of a different historical, philosophical model, which frames the contest between the two positions.

II. Historical Overview

“In the case of every incident of war some time must elapse before its completely detailed and satisfactory history can be written.”

*Michael McKeogh*

Software is a strange intellectual property for many reasons, but largely because the legal discourse and the very technical vocabularies have been in dynamic change since the beginning. Software lacks the historical bedrock of centuries of settled debate common to land, a neighbor’s cow, and the many creative and useful arts. The public, the software developers and the law have tried to wrangle this terra nova into traditional property boxes. Yet, despite the chorus of voices in the total developer ecosystem, two distinct schools of thought arose, each with substantial power to craft and dictate language with authority. No thing has meaning in a vacuum. There is no moral right, nor logical impetus, directing the right fit of intellectual property for software. It is only through the pedagogy of the loudest and most popular voices that software has found its current home.

---

\(^3\) *Id.* at 1021.
Although software has been around since the 40s, large scale development and distribution began in earnest in the ‘60s when “IBM led the way in selling integrated computing packages to large customers.”\textsuperscript{4} These mainframes were massive machines with integrated software and hardware, protected by the contract between buyer and seller. The contracts were negotiated, and subsequent breaches on the contracts were resolved by contract law. Concurrently, software was being protected through trade secret. Both trade secret and contract were limited in terms of protection and remedies, but the social imagining of software was shaping around the technological growth, eventually reaching an unprecedented ease of copying and demands for interoperability.

As software transitioned from hardwired code to something independently distributed, “… paper contract[s] solely governing license of the software gained significance.”\textsuperscript{5} The basic structures and permissions of the licenses were employed to grant rights under copyright. Yet, throughout this time, the copyrightability of software was still in question.\textsuperscript{6} The issue of software’s functionality played no small part in the debate. Somewhat ironically, even the opponents of software copyright have embraced the copyrightability in the current fight against patentability.

The major players in these battles were titans and future titans of the software industry. Their goals were to force this new thing, this terra nova, this new material;


\textsuperscript{5} Laster, at 683.

\textsuperscript{6} Id., at 629.
software, into boxes of protection long established for the competitive field of commercial use. The mainstream model is one derived from deep cultural policies of imperial power. Corporations, and their intellectual property devices, are empowered by grants of power, be they recognition or limited monopolies, from the sovereign. The philosophical matrix that spawned this utilitarianism was, and is, the dominant paradigm for business. Property is protected through power. Control over, and use of, property justifies this power. In ancient times, when the crown ceded power/property to someone, individuals took force into their own hands to protect their property. Whereas today, law is the modern substitution for violence. Entities producing BUS are commonly labeled proprietary or commercial because of this backdrop where finding an entitlement leads directly, and inevitably, to enforcing it to the extent of the law.

By the 1970s, the societal exposure to software reached a critical mass and clusters of programmers, once lone hobbyist and graduate students, began networking on university servers. The revolutionary counter culture of the era produced pockets of thought in the public discourse that deviated from vertical domination processes in favor of horizontal collaboration. From these seeds, what Christopher Kelty describes as a ‘recursive public,’ grew in opposition to the hegemonic force of the dominant culture.

---

8 Christopher M. Kelty, Two Bits: The Cultural Significance of Free Software (2008)
Kelty, in his introduction, says of the movement, “By culture, I mean an ongoing experimental system, a space of modification and modulation, of figuring out and testing; a culture is an experiment that is hard to keep an eye on, one that changes quickly and sometimes starkly.”\(^{10}\) This was the birth of the FOSS community.

To say that what was and has become ‘FOSS’ is merely a developmental and distribution model, grossly understates the meaningfulness that the principles have to the community members. FOSS is a way of life for a pool of individuals who found interconnectedness through sharing their work. Their work, that is, in the sense that work can also be your hobby and joy in addition to a vocation. FOSS is therefor, a cultural identity for many. Quoting David Bollier, Gabriella Coleman writes that free software, “has become an existence proof that individual and collective goals, and the marketplace and the commons, are not such distinct arenas.”\(^{11}\) The philosophy was apolitical, counterintuitive economically, and highly moral by its own standards. This revolution, by its acceptance and transposability, threatens the hegemony of exclusive BUS paradigms.\(^{12}\)

By the 80’s, the issue of copyrighting software was largely settled, with both BUS and FOSS using licenses to dole out their exclusive rights and the Copyright Act to own Hollywood given fame. FOSS, by re-imagining IP protection is another example of a marginalized thought usurping the empire though the use of it’s own tools.

\(^{10}\) Kelty, at 2.


\(^{12}\) Id, at 2.
find their remedies. Early FOSS was doing so, even without consciously intending that to be their transaction model. An example of this is the distribution of Richard Stallman’s EMACS text editor. EMACS was a developer’s tool designed in such a way as to permit and work with user created extensions.  

Richard Stallman didn’t invent the program, but in 1981, after several years of dealing with a confusing mass proliferation of user edited EMACS, he did instigate a major first step to open distribution.

Stallman’s big idea was the EMACS commune. The ‘instructions’ were included as an attachment to EMACS user manual. The document informs users that EMACS is free of charge, but that usage is conditional. The conditions were: send back improvements made, restricting redistribution to original version or improvements separately, and restricting distribution on local systems. Stallman may have envisioned this arrangement as a new form of distribution, but effectively, it was a license conditionally granting copy and distribution rights. Shortly thereafter, EMACS spawned issues of control over visions of free and commercialization. This struggle in turn led to FOSS’s great education in copyright law, resulting in an incredible legally sophisticated community of programmers.

After a great deal of confusion ascertaining what form of creation software is exactly, the courts have recognized software as patentable as a method claim. Software patents are possible because of return by the courts to a first principles approach wherein,

\[\text{Kelty, at 202.}\]
\[\text{Id.}\]
“ingenuity should receive a liberal encouragement.”15 This fundamental approach splits the atom, guiding the separation of patentable subject matter and “law[s] of nature, physical phenomena, or abstract idea.”16 Still, as a method, software is unlike any other method patent because it does exist in a transferable form, and because all steps of the method are in action upon running the software, if not also upon making a copy.

Despite the patentability controversy that yet rages on, statistics show there are over 200,000 software patents and the number grows with more than 20,000 a year.17 Software is being patented. The effect has been as complicated in practice as it was in theoretical debates. BUS software patents are criticized as flooding the field with bad patents. The FOSS approach to these software patents has been either an outright rejection of the idea, activism attempting to invalidate patents, creation of defensive patent pools, retaliation clauses in licensing, or covenants not to sue.18 While these roles at least recognize the importance of addressing patents, they do not make an effort towards realizing any benefits patents might have for FOSS.

15 Bilski v. Kappos, 130 S.Ct. 3218, 3222
18 Michael B. Enschlag, Software Patents: There is Nothing to Fear but Fear Itself or Hills and Valleys of Protection, 989 PLI/Pat 251, 266 (2009). “Most attempts to deal with patents and open source are directed to invalidating patents (invalidators), banding together to oppose patent owners from asserting patents against open source software such as Linux (banders), to restricting the ability of patent owners who utilize open source software to assert their patents (license terms), or to issuing covenants not to assert patents under certain circumstances (covenantors).”
In the post-internet age, the need for collaboration became absolutely crucial to success and networked functionality. May BUS and FOSS principles comingle to allow vital access to source code, enabling the interconnected world to work. Since both paradigms have the goal of lowering costs and raising interoperability, both are in a struggle for ownership of the concept, ‘open.’ Open has cache, and while respected organizations like the Free Software Foundation promulgate definitions of ‘open,’ in the patent context, the meaning of an open patent is still in chaos. Although software owners may enjoy concurrent protection of patents and copyright, this paper is directed at the differences between the two. The stage is set for a battle over ‘open’ in the patent field. So far, BUS entities have been appropriating the term and FOSS entities have been patent averse. In the following sections, this paper will address the rising issues of combining/adapting patents to FOSS principles.

II. Cost and Benefits of Open Software Patents

There are several important costs, benefits and considerations for a FOSS to deal with before they take on defining and licensing ‘open patents.’ Some issues are directly related to a perception of patents as having an inherent value, distinct from, and undesirable compared to, copyrights. Other issues are more tangibly related to cost and legal consequence. Despite these serious considerations, there is no principled reason why the pattern of counter culture activity within FOSS to cease countering trends to lock-up software by subverting the very tools of monopoly.

19 Mann, at 7.
A. Philosophical Considerations

All intellectual property is about control. A copyright, in its exclusive rights, is a limited monopoly. While a true dichotomy exists between public and private property, copyrights are not an especially saintly variety of private property. FOSS, and indeed, the entire industry that uses licensing, knows that a copyright holder’s exclusive rights are not superfluous simply because they may be licensed away. The licensing of these rights is, many times, what enables the preferred forms of remedy.\textsuperscript{20} The distinguishing mark between patents and copyrights is that FOSS has experienced a systematic desensitization after spending so much time copyrighting software. Patents, on the other hand, receive a very bad rap with the FOSS community.

Patents are no different from copyrights in that they both convey control.\textsuperscript{21} The breadth of the rights that they control differs statutorily, but just as in copyright, there is nothing compelling a patent holder to exercise their rights to the exclusion of others. Patent licensing has a long history of parsing up patent rights according to the patent holder’s business needs. A patent holder would not be forbidden, by any rule, from giving away all or most of their rights in order to comply with FOSS. If it is not in the underlying rights involved, then perhaps there are other reasons for the venom towards

\textsuperscript{20} Injunctions in both copyright and patent require the license to have a nexus with the respective exclusive rights of the IP being licensed.
\textsuperscript{21} Ius mentis, Law and Technology Explained, http://www.iusmentis.com/computerprograms/opensourcesoftware/patentrisk (Last Checked April 2011)A simple internet search of the terms ‘patent’ and ‘open source’ will produce a deluge of responses like this one: “Patent and open source software are fundamentally incompatible.” Yet, for some reason, likely attributable to exposure desensitization copyright no longer invokes this ire.
patents. Regardless of the social politics or aversion to the word patent, it has not gone unnoticed that patents can supplement the gaps in protection left by copyright, especially in light of the fact that the other means of software protection, trade secret, is not particularly useful to upholding the goal of sharing information.22

There is a false dichotomy between FOSS and BUS philosophical underpinnings. Both are grounded in utilitarian theories with an aim to “ensure the proper incentives for the production of intellectual properties.”23 The word ‘patent’ triggers associations with monopoly and control, as is the common vernacular to any discussion on patents. These words themselves echo labels used to describe things that are not FOSS; proprietary and commercial. However, “FOSS is protected by a proprietary right, namely copyright law.”24 Likewise, FOSS is used in many commercial projects, even if they derive profits from means other than royalties.

There is a deeply rooted ideological fear of patents in the FOSS community. Viewing the rivalry of transactional models as a form of cultural warfare, a FOSS adoption of patents is the equivalent of becoming the enemy. However, an absolute rule against patents raises the commonly discussed issue in forums and blogs of the survival of open source in world rapidly turning to software patents. Blogger and FOSS advocate,


24 Gomulkiewicz, at 1020.
Florian Mueller, thinks the end of FOSS caused by patents is simply a change to FRAND royalties.²⁵ The reasoning is that since that FOSS has proven flexible enough to endure numerous non-royalty commercialization, then FRAND, perhaps at a zero rate as well, preserves the spirit of FOSS by not throwing out the baby with the bathwater. Like Meuller, Nigel Howard sees a reasonable royalty rate as a normal solution, partly because on a suit for infringement, the reasonable royalty rate for FOSS would be a zero amount anyways.²⁶ Howard also see FOSS patents as delivering benefits on more conventional way, albeit, in ways that BUS patents are presently conducting business. The large looming fear is also that patents will block the field and open source copyrighted material will be cast away as inevitably infringing code, but continuing to produce open source software, patented or copyrighted, will only improve software patents by propagating prior art.²⁷ Additionally, a software patent, one that just happens to be held by a FOSS is not necessarily an open patent. Instead it could be a bartering chip for a cross license with a BUS entity. In that case, the open source community could “enjoy immunity from the proprietary software company’s patent.”²⁸

B. Social and Economic Costs and Benefits

1. Costs of Patents as a Deterrence

---

²⁶ Howard, at 2.
²⁷ Id.; Mann, at 29.
²⁸ Howard, at 2.
Outside of a philosophic aversion, there is a practical deterrent to anyone, not just FOSS, seeking a patent. Whereas a copyright is free, patents are costly.\textsuperscript{29} A short road map to a software patent starts with the first hurdle; does this software meet the high demands of a patent? This question should compel an inventor to evaluate his invention and make the strategic decision to copyright or patent. The time limit of development could spoil the patentability.\textsuperscript{30} They must further be honest with themselves about the novelty and non-obviousness of their invention. These concerns are simply not considered in a copyright preparation and so FOSS would necessarily be forced to prepare for a new major undertaking in their development. A community of developers accustomed to sharing and reusing ideas may find that only a small portion of their software is even suitable for patent protection, but even once the right fit is reached, then there is the application process.

FOSS members, having spent years wrangling with copyright and licensing, are a legally sophisticated community. FOSS is substantially a do it yourself, development model. Therefor, the practical reality of hiring patent attorneys must rile suspicion. The cost of legal fees are further injury to the insult. IPWatchdog founder, Gene Quinn, writes that attorney fees for a relatively complex invention, such as business methods typically range from $12,000-$15,000.\textsuperscript{31} For highly complex inventions such as networking systems, $15,000 is merely the base figure. Then there are the government filing fees

\textsuperscript{30} 35 U.S.C. § 102 (b)
\textsuperscript{31} Quinn.
which increase with the number of claims in the application. The instinct to cut costs by making a simpler claim cuts against the strength of the patent and is not advisable.

FOSS may have a better time with the patent search process than other developers. As software patents were initially filed, the search was a frustrating ordeal because there were no previously filed software patents, and the field of copyrighted works is necessarily vast. FOSS developers however, have an advantage because the interconnectedness of their development community can search on a project far more efficiently in their numbers than a several thousand dollars can get from an attorney.

Quinn estimates that the cost for a provisional patent application for software is “typically $3,000 plus the filing fee and drawing costs.” With other costs added together, patent search, provisions application, filing fee with USPTO, nonprovisional application based off of provisional filing, and illustrations is roughly $17,110. This cost only goes up. Does this mean that software patents are not worth the trouble? The substantial trend of BUS patents would seem to indicate otherwise. For BUS, the benefits outweigh the cost, and even without royalties, it is also quite likely that the same can be true for FOSS.

32 Id.
33 Id.
34 There is always a patent statute dedication to the public. An extreme preservation of software freedom, in cost at last, but also in access would be to truly dedicate the patent to the public, an act not possible under copyright. “A patentee, whether of the whole or any sectional interest therein, may, on payment of the fee required by law, make disclaimer of any complete claim, stating therein the extent of his interest in such patent.” 35 U.S.C. § 253 (2006); Mutschelknaus, at 430.
2. Open Patents to Promote FOSS Principles?

Across the internet, the benefit of patenting software is stated inaccurately as: ‘a copyright only protects your expression, but a patent protects the idea.’\(^{35}\) In every lesson on copyright, the phrase, dichotomy of idea vs. expression is raised to expel any notion that copyright protects ideas. It is absolutely clear that copyright only protects the expression. The analytical framework of filtration-comparison is applied by courts to separate pure idea elements from protectable portions.\(^{36}\) For software, the expression is only that arrangement of code, or the nonfunctional design elements. For this reason, many commentators recite that copyright is a limited protection in comparison.

What is it that FOSS really wants to protect? The mission of the Open Source Initiative states that “[o]pen source is a development method for software that harnesses the power of distributed peer review and transparency of process.”\(^ {37}\) The rest of the mission statement proclaims that the organization is to educate and advocate for open source. The Free Software Foundation describes itself in a similar way, promoting a value system where, “free software has become the foundation of a learning society where we share our knowledge in a way that other can build upon and enjoy.”\(^ {38}\) Taking these principles together, if given a choice, FOSS should prefer a form of protection over

\(^{35}\) Again, a simple internet search lands hundreds if not thousands of examples of this simple misstatement. The gist of the statement is a well meant attempt to articulate the breadth of a patent over that of a copyright. However, a patent may not be acquired for an idea in the abstract, so clarity here would alleviate misunderstandings


the broadest manifestations of ideas and ideals embodied in the software over mere protection for the arrangement/ expression of that idea.

Although you can not patent an idea, the steps making up a system, method or process are not limited to such a narrow exact combination or sequence of integers. A software patent’s claims, wide or narrow, cover more materials than a copyright. Even further, whereas a copyright infringement requires a showing of actual copying, under the doctrine of equivalents, a patentee’s protective net covers non-literal copying.\textsuperscript{39} If the principle is to ensure that the ideas embodied in a FOSS work are freely available, a patent would stop far more BUS competitors from writing around a copyright. It is true that a patent could also be ‘written around’ but at much greater difficulty.

C. Legal Considerations for an Open Patent

Even if a FOSS patent can be harnessed to defend FOSS principles more broadly than copyright, the next issue is to look at how that protection would look in court. Back to basic principles; a patent gives a patent holder more and different rights compared to a copyright holder. This may reek as an obvious statement, but make, use, sell, offer to sell and import are broader than the exclusive rights of copyright. The significance of this fact has not been lost on the software industry, as a whole, as seen in the migration towards patent protection.

\textsuperscript{39} 35 U.S.C. § 112 ¶ 6
So you copyright a piece of software. Suppose a competitor without access to your software independently writes the same code. You have no cause of action against them. You would under patent law. Another competitor, having access to your code, hires clever developers and achieves the same steps as your program does, but through a different arrangement of lines of code. You have no cause of action for infringement under copyright, because of the idea vs. expression paradigm. You would under patent law, because of the doctrine of equivalents, as well as, for any, non-copying, literal infringement meeting the claims. A competitor could also use your software without violating copyright in a way that would have violated a patent. The only flip side is that if a competitor wanted to distribute the software free of charge, you would have an infringement action under copyright, but possibly not under patent. The exclusive rights between patent and copyright have some overlap, but for a community like FOSS, so immersed in copyright, the significance of the differences may not be immediately apparent.

1. Comparison of the Rights under Copyright and Patent

---

40 Procter & Gamble Co. v. Colgate-Palmolive Co., 199 F.3d 74, 77 (1999). Independent creation is an affirmative defense to copyright infringement.
42 Even if the copying was identical, an infringer may have a fair use defense under copyright, but not under patent.
43 Some legal considerations are not unique to copyright law, such as misuse. Whether or not open patents are a restraint on trade is highly dependent on the circumstances, but the FOSS licenses are outside of the usual license misuses: such as tying, grant backs and potentially anticompetitive geographical deals. Herbert Hovenkamp, *Antitrust*, n. 27, 98, 185 (4th Edition 2005). FOSS may articulate substantial precompetitive effects to ameliorate any misuse or antitrust concerns. Howard, at 3.
a. Make = Copy?

In the Patent Act, § 271, direct infringement does not explicitly define what it is to ‘make’ an invention. It can be inferred from cases that to ‘make’ is to manufacture an invention that contains all of the claims of the patent, or when “uniting all the part of the invention into a operative whole”.\(^{44}\) The act of “making alone, even if none of the other listed activities occurs” is enough to prove infringement.\(^{45}\) In the software context, at first, this does not seem different from ‘copy.’ However, copying requires showing access and substantial similarity.\(^{46}\) Further, the copyright infringer may have a defense of fair use. In patent, direct copying is not necessary. Make is a very, matter of fact, determination that the accused invention meets the patent claims.\(^{47}\)

b. Use =?

Use is not an exclusive right of copyright. This has been a major fact in copyright license cases.\(^{48}\) Yet, just as in making, using alone is grounds for patent infringement, where use is when someone employs the invention “for its underlying beneficial purpose or function.”\(^{49}\) This infringement applies even to “one who buys an infringing device from another.”\(^{50}\) For FOSS, this means that every user, in using, must comply with the

\(^{44}\) R. Carl Moy, Moy’s Walker on Patents, MOY-PAT § 14:1 (2010).
\(^{45}\) Moy, at § 14:19.
\(^{46}\) Arnstein v. Porter 154 F.2d 464 (1946)
\(^{47}\) There is some question as to how method claims can meet the making element of infringement. However, software is unique that all of the operative steps can exist in a quasi-static form when the program runs.
\(^{48}\) National Car Rental v. Computer Assoc., 991 F. 2d 426 (1993). A suit on a license purporting to limit use was found to have not been preempted by copyright law, and instead was limited to contract law.
\(^{49}\) Moy, at §14:32.
\(^{50}\) Id.
standards and terms of a FOSS license. This degree of power to enforce a compulsory freedom over any operation of a program is an untapped reservoir of power to propagate FOSS conditions.

There is a patent defense for experimental use, which ultimately fits nicely with FOSS principles of freely permitting development. This limited right to use, even without a license, is predicated on the use not interfering with the patentee’s right to “commercially exploit the invention solely.” 51 In few industries outside of software are there actually independent tinkerers, working on software for purely “gratifying curiosity or a philosophical taste, or for mere amusement.” 52

c. Sell = Distribute?

Sale is a more difficult issue. In the common meanings, the copyright word ‘distribute’ is a broader term than patent’s ‘sell.’ Sell is determined from an underlying contract, “when a contract has been executed that transfers title, or some other possessory interest…” 53 This may exclude licensed transactions conveying less than possessory interests. FOSS must take heed of this issue as will be addressed in the section on licenses. Although Moy goes on to say that cases dealing with the license issue do not distinguish between a sale and a license for infringement, the problem looms. The distinction between sale and license is litigated in exhaustion cases. The cautious approach may be to simply be aware that the ability to restrict distribution outside of

51 Moy, at § 14:1.
53 Moy, at § 14:38.
sales, may be curtailed by the word sell, but still overall stronger through make and use rights.\textsuperscript{54}

d. Development Problem – Public Use Bar

Another issue, one which FOSS practices may be unsuited for, is that their pre-patent application development of software could bar patentability. FOSS works because thousands of contributors work, without compensation, sometimes for a very long time refining a piece. Linux is a prime example. Linux is based on an early series of Unix operating systems, first developed in 1969. Incremental changes from ’69 to 83 led to the release of GNU by Richard Stallman.\textsuperscript{55} Utilizing many of these improvements, Linus Torvaldes released Linux in 1991. From ’91 to present, a vast number of programmers have tinkered with Linux. Some of this time can be characterized as beta testing, but Linux, the program itself, excluding its various components, is not patentable because it has been in public use for longer than a year.\textsuperscript{56}

In patent law, under § 103(b), a patent may not be obtained if the “invention was … in public use or on sale in this country, more than one year prior to the date for

\textsuperscript{54} Id., “The concept probably does not apply to inventions characterized as methods.” Again, like in ‘make,’ there is an issue with method inventions. However, software can be conveyed in a packaged form, different from most method inventions, and logically, should be able to be infringed by a sale.


\textsuperscript{56} Linux, the open source project as a whole, does hold many patents. This example is primarily for the long timeline leading to collaborative, incremental development. Operating systems are often comprised of numerous patentable software components.
application for patent in the United States.”57 The courts have established an experimental use exception. In example, one court case, a road that was used for more than a year by the public was still experimental use for the inventor to refine the surfacing technology.58 FOSS must be able to characterize the duration of essentially public use as beta testing. The factors involved appear a logical fit to the development of software, with the exception that the duration of time must be cut off at some point by the core development group when they feel it is time to patent the invention.59

The patent benefits for FOSS are superficially the same benefits that a binary use patent enjoys. So long as FOSS is aware of the change in their rights from copyright protection, the community may self educate along the same lines as it has in the past to become sophisticated copyright users. Knowledge of patent differences, in itself, is insufficient to instruct one on drafting licenses for that patent. Holding a patent gives the patentee the right to sue for infringement if anyone out in the world makes, uses, or sells an infringing invention. FOSS infringement suits usually arise out of a somewhat different context. The FOSS works are licensed without prejudice to anyone in the world. Infringement occurs when that licensee does something with the software that is outside of the license. The breach of license terms can be divided into two categories; breach of

57 35 U.S.C. § 103(b)
59 These factors include; the nature of the invention, the length of time over which testing occurred, the complexity of the invention, the level of supervision or control maintained by the inventor over the tests, obligations or limitations placed on users during experimentation, whether feedback was received by the inventor, the environment for which the invention is intended, and the testing environment, and can not be merely for testing market acceptance.
terms which condition the grant of intellectual property, and breach of terms outside of
the IP but important to the deal.

2. Covenant or Condition

Several copyright cases apply by analogy to distinguish a covenant from a
condition. In the former, a breach will trigger contract law, while in the later, copyright
will preempt state contract law, and statutory remedies are attainable. The material
difference in available relief for FOSS is in the injunction under copyright or patent law.

In Jacobsen v. Katzer, Matthew Katzer, CEO of a software developing company,
was accused of securing patents on open source software licensed from Robert Jacobsen
and the open source JMRI Project through the Artistic License. Jacobsen sued Katzer
for declaratory judgment of the unenforceability and invalidity of the patent, and also,
importantly for this point, for violation of copyright and breach of contract. To have a
claim under copyright law, Jacobsen’s contract terms had to have a nexus to a copyright
subject matter in the articulation of the breach of the conditions of the Artistic License.

The clauses of the Artistic License pertaining to attribution are linked to the
exclusive right to distribute, because attribution “enables the copyright holder to retain
the ability to benefit from the work of downstream users.” Additionally, clauses
pertaining to altering the software related to the exclusive right to create derivative

60 Jacobsen v. Katzer, 609 F.Supp.3d 925 (2009). The other important lesson from
Katzer is the availability of injunctions for open source projects.
61 Id. at 929.
works. The take away lesson from this case is that injunctions are possible for FOSS, but the wording of the license can make a tremendous difference. A license that fails to condition the granted permissions could unintentionally cause a licensor to lose the benefit of their bargain.

3. Drafting Solutions

a. Condition on Sale

A software patent license with a grant conditioned on attribution may not affect the sale of the software. The Katzer reasoning, that attribution lets the “downstream users know about the collaborative effort to improve” the software project is still present, but whereas FOSS copyrights can characterize this as effecting distribution, sale is a much narrower concept. This is a problem because many FOSS projects never involve a sale and are freely licensed and downloadable. An open license with express language for sale conditions would trigger patent law, but it would also be a failure as a license of general applicability to the FOSS community. If sale is characterized broadly to include licenses as Moy defines it, as “when a contract has been executed that transfers title, or some other possessory interest…” then, the license contract could be interpreted in an unanticipated, possibly problematic, way.64

63 Id, at 1379.
64 Moy, at § 14:38.
65 Certainly, a license could condition permitting only free distribution and no sales of the invention would be a valid condition. However, this is not the meaning of ‘free’ in FOSS. The Free Software Definition from the GNU Project, is “Free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not “free beer.” http://www.gnu.org/philosophy/free-sw.html
b. Condition on Use

And how would a condition on the right to ‘use’ look in practice? A copyright license that merely conditions terms upon which licensee may modify and distribute does not reach ‘using’ the invention ”for its underlying beneficial purpose or function.” A new license grant could be drafted that conditions any operation of the software as use dependent on some obligation upon start up. However, that sounds impracticable and not easily bent to serve FOSS principles. Use therefor, is more useful in finding infringement of unlicensed works that infringe on the patent and are being licensed for fees by a BUS competitor rather than finding infringement against a licensee for breaching license terms.

c. Condition on Make

A condition on ‘make’ is likely the strongest position for FOSS to take. Software is made again with every copy, as in the phrase, to make a copy. A condition that is raised anytime one “[unites] all the part of the invention into a operative whole,” is the penultimate moment in software’s existence where a condition has the most force. The license could then read, ‘Any time the program is made or copied, the terms and conditions of this license must be included.’ This preserves infringement actions against unlicensed independent inventions that are made without the terms for any purpose,

66 In an effort to argue that the license is a sale for the purposes of establishing infringement of sale, then the licensee could latch onto this as a moment of patent exhaustion and possibly be free from the conditions of the license.
67 Moy, at § 14:32.
68 Moy, at § 14:1.
unless they conform to the license. Since the most common form of software distribution is making copies, the breadth of copyright’s exclusive distribution is mirrored in the patentee’s exclusive right to make or condition a licensee’s ‘making’ of the software.

III. ‘Open’

On the front lines, the common usage of ‘open’ shift and mutates as all participants stake their claim by attaching the word to their projects. This causes a great confusion for anyone desiring to contribute or make use of these projects. Keeping the core principles of open source movement as a starting point, the vast forking has open projects that deviate so far that there is little more than a marketing buzzword to the usage. Others, in fields outside of software, have adopted the word in ways very close to the original intent. The problem of language slippage is substantially compounded as the word is applied in the patent context. The assortment of rights and obligations part and parcel to open copyrights is being haphazardly applied to cover all IP in licenses that fail to mention patents. In others, patents are explicitly mentioned, but in ways other than license grants. The few that are express license grants muddle language of exclusive rights from copyright and patent.

The incongruous fit between things that are labeled open and patents suggests that software patent’s harshest critics are right when they say that the two are fundamentally
opposed. This however overlooks several major points: people are profiting from the confusion, the bad fit in licenses can be resolved.

The simplest definition of Open Source Software, from PC Magazine Encyclopedia, is Software that can be modified and recompiled by the user. The usual, conventional core meaning has four essential elements: access to the source code, the right to run the software for any purpose, the right to change the software in any way, the right to redistribute the original software and any derivatives. The more complex definition from the authoritative and influential Open Source Initiative has 10 factors that add anti-discrimination rules, viral licensing, and limitations to tying. Against this backdrop, if a scenario purports to offer open software, how many factors can it drop before open is a meaningless term?

The linguistic effect, which Coleman explains drawing from philosopher Jacques Derrida, as a “naturalize proposition (like homosexuality) or commonsense social fact both presupposes and ultimately propagates what it excludes.” Stallman and FOSS generally, naturalize a use of copyright, which presupposes the traditional uses of the

69 OSS Definition, PC Magazine Encyclopedia, http://www.pcmag.com/encyclopedia_term/0,2542,t=OSS&i=48648,00.asp (Last Checked May 2011).
70 Mutschelknaus, at 409.
72 Coleman, at 3.
intellectual property right itself, by securing protection, and giving much of it away. It then becomes a question of authority. The law, both arising from the Constitution and from copyright and patent statutes, holds a significant authority, but language and actual practice shapes usage and actual meaning. FOSS, by using the law, has created a subversive alternative outside of the courts. Movements outside of FOSS, entities outside of the philosophical groundings, also have the same power to speak with authoritative voices, creating meaning. Only time will tell what ‘open’ will mean for patents.

A. Open Licenses Touching Patents

More than a few FOSS licenses mention patents, but mentioning is not quite granting an open patent license. Lawrence Rosen writes that, to comply with FOSS, “a patent license necessary to make, use, or sell the software under the license must not prevent the creation of derivative works or prohibit use anywhere in the world.”73 As aspirational as that sounds, perhaps an open patent license should start with a grant and conditions relating to the exclusive rights of a patent holder. After examining a few popular FOSS license’s patent terms, the problem of patent grants seems to be fairly consistent.

1. GPL

- Basic Permissions. (edited and italicized for this paper)

All rights granted under this License are granted for the term of copyright on the Program…

---

This License explicitly affirms your unlimited permission to run the unmodified Program.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this

Termination

Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

Patent Provision

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. […]
Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version. 74

The General Public License is now in its third incarnation.75 The guiding force behind the GPL is Richard Stallman, a vocal advocate against software patents. The license therefor is primarily a copyright license that “makes creative use of the copyright monopoly by permanently giving away the exclusive rights of a copyright holder.”76

The GPL 3.0 does address patents, but not in a way that presumes that a patent license is actually the goal. This becomes apparent in the definitions section, which defines copyright, but neglects patents. Even the definition of Program is fixated on a program being a copyrighted piece of software. The basic permissions identify the primary rights being granted are copyrights, but then goes on to spell out the rights to make, run and propagate. Make is clearly a patent right. The right to run a program is akin to the patent right to use. Only propagate relates back to the copyright exclusive right to distribute. If these rights being granted are in anticipation of a patent, or dual rights, then why only name copyright in the license?

75 Rosen, at 103. Though details have changed from 1.0-3.0, the nature of the deal is still that, “You may have this free software on condition that any derivative works that you create from it and distribute must be licensed to all under the same license.”
The traditional mention of patents in open licenses is still there in the termination clause. However, the actual patent provision is more interesting because it does purport to grant an open patent license. The correct rights are included as make, use, sell, offer for sale, and import. FOSS principles are also included in rights to modify. The strange thing is that there is still a foot out of the door because of how a patent comes into play. Under the GPL 3.0, the patent provision is triggered when a contributor to the project contributes a copyrighted program, which has some patent protection to it. Only patents essential to the copyrighted contribution are covered by this licensing model. How then would one contribute a patent itself? The answer is unclear.

A covenant not to assert a patent is one way to express what a license is. Another way to think of a license is a permission to do something with the patent. When a patent grant is added only because it is implicated in the copyrighted contributions, the permissions should be spelled out in patent terms. Yet, the principles of FOSS, including derivative works coming from copyright practices, should be considered. For example, the MozPL and IBMPL both “… grant royalty-free licenses to patents that cover the licensed software in return for requiring the licensee to grant licenses to any patents it

---

77 Any attempt to enforce patent rights on an invention containing GPL licensed components certainly is an act counter to the entire scope of the license and the FOSS principles. Logically, the same should be true of any attempt to enforce copyrights on the same.

78 Additionally, the license only applies to those who distribute the program unmodified and those that were licensed under the GPL 3.0.

79 The Apache License and the Creative Public License have the exact same problems. In the Apache license, the definition of Contribution is work submitted by the copyright owner. The Creative Public License also only has a patent catchall provision that is triggered only if the “Contribution causes such combination to be covered by the Licensed Patents.”
receives covering improvements to the software.”

Even though most existing open licenses dealing with patents expressly narrow the patent grant to the contributed patent and therefore reserving rights to improvements, there is no reason, when dealing with a FOSS, to think that the improvement represents threat of an infringement action. After all, a FOSS is likely to license that improvement patent on the same terms.

While recognizing the contemplation or patents present in these licenses, there appears to be a disconnect. It is as if the patent clauses are afterthoughts, or contingencies should a patent be implicated by copyrighted contributions. Such an approach presupposes that the inventor does not want to grant patent rights, or that the default position is to only secure a copyright. Mixing the two forms of IP in one license is not inherently problematic, but the copyright genesis of the FOSS licenses has left a clutter of copyright language. In order to have the best chance of success enforcing the license, it goes without saying, a license should use the most accurate grant language corresponding to the IP being licensed. However, the opposite route appears to work fairly well with the BSD.

2. BSD

The Berkley Software Distribution license is “designed to permit the free use, modification, and distribution of certain University of California software without any return obligation whatsoever on the part of the licensee.”

The purpose and goal of this license is getting the software into the hands of the community, and that directive ignores

---

80 Howard, at 2.
81 Rosen, at 73.
the nature of the IP protection on the work. Instead, the grant is a very broad, bare license to the software, whatsoever protection it may have on it.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.⁸²

The problem is that the unwary may accidentally convey patent rights in with their copyrighted programs in the form of an implied license. The solution has been the creation of a variant license called BSD Clear. The additional language narrows the broad grant from all IP, to all IP except patents.⁸³ This approach recognizes that licenses to patents and copyrights might best represented with specifics, rather than unspecified promises not to assert patents. The issue isn’t the enforceability as much as it is about the clarity of permissions that the FOSS is really granting.

B. Pledges

1. What are Pledges?

---

As FOSS gained acceptance as a legitimate transaction model beyond initial public perception of hackers, communists and thieves, BUS entities began to turn their heads. It is now not at all uncommon for BUS to utilize substantial portions of FOSS projects in their products and services. Another way that BUS has engaged FOSS has been contributing to the projects as angel investors and thereby earning credibility and goodwill with the FOSS developers. A third way is also arising, and it is far less clear if it supports or actually detracts from FOSS principles. Major BUS producing entities, rather than draft open source licenses, have issued public statements in the form of non-assertion pledges. The pledges do not go so far as public dedication of the programs. Instead, those taking advantage of the pledge tread in nebulous territory somewhere between a license and a gratuitous, and ephemeral statement of intent to not enforce IP rights at this present moment.

IBM, Microsoft, and Sun have all issued patent pledges. Sun has offered patents pertaining to the Unix based Solaris operating system. Microsoft promised not to sue for some of it’s standards and interfaces. IBM, famously, led the charge by pledging not to sue for 500 patents. As friendly as these pledges appear, skepticism abounds about how far to trust this gratuitous stated intent not to sue. The pledges are not unconditional public dedications. Instead, “[w]hile mak-ing public promises of non-enforcement, these companies do not actually disclaim their patent rights with the USPTO. Absent the

---

84 Mann, at 30, “Fur- ther, the underlying technology is not available for the development of proprietary offerings by competing products or services firms (such as Microsoft and Apple, both of which have used UNIX technology in their operating systems). Nor is the grant absolute, because it is not effective against a firm that asserts patent claims against IBM.”
elements of a contract, the enforceability of these promises is not entirely clear.\textsuperscript{85} For the pledge to be a contract, there must be the usual: offer, acceptance, and consideration. Posting the pledge publicly may be the offer, and taking advantage of the patent is acceptance. Jacobsen v. Katzer has demonstrated that the contractual requirement of consideration can be viewed in light of totality of the deal, and not strictly a literal exchange of cash or a peppercorn. But is a pledge a license contract? The issue-ers of the pledge likely think not, but it may be an implied license nonetheless.

No contract between the parties is necessary for an implied license, which instead draws upon estoppel.\textsuperscript{86} The conduct and wording of the pledges indicates that one has the permission to do something with the patents. In implied licenses, the “expectations of the targeted user group…” determines what exactly those permission are. Id, at 421. A FOSS user of the pledged program would have reasonable expectations derived from their ordinary understanding of the four essential freedoms of FOSS. After all, the pledges are adorned with the paraphernalia of ‘open,’ and this expectation may override the pledger’s attempt to redefine industry usages.

What then, should the scope of these implied licenses be? Most all pledges purport to condition uses and sales. Yet, a FOSS member’s expectations would say that field of use restrictions are prohibited by Open Source Definition.\textsuperscript{87} What about when a combination with the ‘open’ patents and new product, and new product is sold? Should

\textsuperscript{85} Mutschelknaus, at 413. 
\textsuperscript{86} Id. 
\textsuperscript{87} Id. *425.
an implied include a permission in this situation due to the nature of software’s common implementations?\textsuperscript{88} The answers are fare from clear as actual situations are arising where FOSS has tried to use pledged patents.

2. IBM and the 500 Patent Pledge

In January of 2005, IBM published the “IBM Statement of Non-Assertion of Named Patents Against OSS.”\textsuperscript{89} In this document, IBM pledged 500 patents to the open source community. The language of the pledge vacillates between broad policy objectives and narrow definitions for implementation. The IBM definition of ‘OSS’ is any computer program whose source code is published, available for inspection and use by anyone, and permits copy, modify and distributing the source code without royalties.\textsuperscript{90} Access to the source code is the first element of most all definitions of open. It goes on to state that the source code is to “be available for inspection and use by anyone.”\textsuperscript{91} This, too, corresponds with OSI requirement that the software be available, without prejudice. Still in line with typical open source software, the pledge impliedly grants the right to sublicense, so long as the sublicense “permits recipients to copy, modify and distribute” the program’s source code. In itself, that looks like the exact same verbiage as an open

\textsuperscript{88} Id. at 426, Mutschelknaus is somewhat incorrect when he says of open source licensing, “In this situation, a patentee would argue that she did not consent to selling the new product because the license is designed to keep derived works in the public domain.” This is a common misperception of FOSS. Requiring that a subsequent combined product be licensed under an open source license is not at all placing the work in the public domain.


\textsuperscript{90} Again, for a patent pledge to offer patents, the language of the definition for how it can be used is copy, modify (derivative works), and distribute. These are copyrights.

\textsuperscript{91} Id.
source license. However, there is no grant language, and the definitions pertaining to copyright rather than patents, creates doubt as to what one can do with, or charge to service, the resulting programs.

The pledge simply does not read like a license contract. “At first blush, there does not appear to be any consideration, as IBM is not getting any cognizable benefit for its promise.”92 Under the Katzer reasoning, open-source licenses have consideration in market share, and non-obvious benefits. But what are IBM’s benefits? If it is an implied license, what to what rights are participants entitled? When the pledge was issued, it generated speculation in the press and blogosphere. The questions were of a slightly different nature: is the definition of OSS compatible with industry usage? Are the rights being conveyed sufficient to enable open purposes? What exactly are the rights being conveyed? What are the real reasons IBM released these patents and does it matter? Richard Stallman’s opinion is that pledges, including IMB’s are leveraging tools for corporations to “point to its own small gestures as “proof” that software patents will not be devastating to free software.”93 The conclusion that a pledge an illusion is perhaps of some merit in light of the Hercules project struggle with IBM.

3. IBM and Hercules

Hercules is an open source project that allows IBM mainframe applications to run on non-mainframe systems. The project was started by Roger Bower in 1999 and has

---

92 Mutschelknaus, at 430.
gained respect and acceptance from the FOSS community. Hercules’s documentation holds itself out at a “community of “IBM mainframers” who enthusiastically support the computing paradigm that has provided their livelihood.”94 The community attitude speaks to developers that like the IBM ecosystem, not hackers bent on taking on the giant. Hercules further, in detailed readme files, directs users to secure the appropriate licenses from IBM to use the Hercules complementary emulator.

In 2009, Bower started TurboHercules, as a “commercial entity exclusively dedicated to Hercules.”95 TurboHercules offers a wide range of mainframe services at low cost. The difference from Hercules is that while Hercules is a development tool, TurboHercules is marketed to corporate and educational institutions that need professional support, training, consulting in addition to functional services of the data. TurboHercules has received substantial support and investment from a range of sources, including Microsoft Corporation.96

In April 6, 2010 the shot heard ‘round the web was that “IBM breaks the taboo and betrays its promise to the FOSS community.” Respected voice in the FOSS community, Florian Mueller, broke the story which then inflamed the blogosphere. IBM’s offending action was sending a letter to TurboHercules indicating that Hercules

---

96 Id.
project infringes on, at least, 106 IBM patents. This, in itself, triggered a response that IBM was attacking the FOSS community. The issue was further exacerbated by the fact that a number of the named patents are listed on the 500 patent pledge.

Mueller characterizes IBM’s stance towards FOSS as “[ending] where its business interest begin.” In this view, the patent pledge only grants a license to philosophical experimentation and nothing more. Perhaps that was IBM’s intention by including language restricting the permissions to noneconomic use. However, within FOSS, the intended audience of the pledge, the economic restrictions on royalties are commonly eschewed in favor of a community standard accepting FOSS compatible commercialization of services. It is unclear if the commercialization was the problematic point for IBM, or whether TurboHercules somehow fails to fit within IBM’s definition of OSS. Mueller, quoting the Wall Street Journal posed the question, “if TurboHercules doesn’t qualify, who does?”

Some of this reactionary response is hyperbolic. Only two of the named patents were on the pledge. Bower’s suggestion that the other 104 patents be added to the 500 included in the pledge is an ambitious request, but why should IBM give up IP that they don’t want to be open? It is unclear if IBM is going to move forward from writing a letter to instigating a patent infringement suit, just as it is unclear if TurboHercules will file for

---


98 Id.

99 Id.
a declaratory judgment. As of July, 2010, TurboHercules has filed for and received an antitrust investigation into IBM’s mainframes with the European Commission.\textsuperscript{100} As this drama unfolds, it is becoming clear that IBM’s idea for an open patent pledge simply does not comport with the expectations of the FOSS community. The use of the word ‘open’ in the pledge, with the hallmark language of a FOSS-like grant, sets up reasonable expectations within the community. However, since FOSS has not propagated a FOSS patent with acceptable definitions and conditions, then out-side of FOSS entities are going to try to name the baby and possibly get it wrong.

VI. Conclusion

While the trend towards patenting BUS software does not necessarily create pressure for FOSS to adopt the practice, the natural incentives of a software patent should itself be utilized to serve FOSS principles. In fact, “pure open source firms increasingly are acquiring their own patents, primarily to protect themselves from the threat of litigation.”\textsuperscript{101} Patents for protection from other patents are one thing, but there appear to be strong incentives to patent for the strength of licensing and upholding FOSS principles against infringers. The longer that FOSS merely dabbles with patent defense language in their licenses, major players in the primarily BUS industries are co-opting the language of ‘open’ for their patents. The conflicting usages of ‘open’ dilutes the FOSS message and


\textsuperscript{101} Mann, at 43.
may impair true FOSS patent acceptance in the future. If FOSS should adopt a more vigorous patent licensing policy, much greater work needs to be done to bring copyright licenses and patent defense licenses into the helpful, user friendly, standard which FOSS has come to represent.