

CLAIMING COPYLEFT IN OPEN SOURCE
SOFTWARE: WHAT IF THE FREE SOFTWARE
FOUNDATION’S GENERAL PUBLIC LICENSE (GPL)
HAD BEEN PATENTED?

Greg R. Vetter[†]

2008 MICH. ST. L. REV. 279

TABLE OF CONTENTS

INTRODUCTION.....	280
I. BUSINESS METHOD PATENTS AT THE INTERSECTION OF SOFTWARE AND LICENSING.....	286
II. CLAIMING COPYLEFT: FOSS LICENSING UNDER GPLv2 AS A PATENT PROTECTED ACTIVITY	288
A. A Hypothetical Claim for the GPLv2 Patent.....	289
B. Utility.....	291
C. Objective Disclosure	292
D. Novelty and Statutory Bars.....	293
E. Non-Obviousness	294
1. <i>Differences Compared to the Prior Art of Software Licensing</i>	295
2. <i>Secondary Considerations: Particularly “Unexpected Results”</i>	296
III. CLAIMED COPYLEFT: ENFORCEMENT MOMENTS IN THE FOSS MOVEMENT.....	298
A. The FSF and Copyright Enforcement of GPLv2.....	299
B. Abhorrence of Patents for Software versus Abhorrence of Patents for Licensing Methods.....	301
C. Non-Infringing Attribution-Only FOSS Licensing	304
D. Some Potential Enforcement Moments for the GPLv2 Patent	304

[†] Assistant Professor of Law, University of Houston Law Center; Co-Director, Institute for Intellectual Property and Information Law (IPIL); biography available at <http://www.law.uh.edu/faculty/gvetter>. My background includes a Master’s degree in Computer Science and nine years full-time work experience in the software industry. My thanks to Katherine A. Franco for her research assistance. For helpful comments and discussion, I thank Oren Bracha, John Golden, Craig Joyce, Robert Gomulkiewicz, and participants at the 4th Annual Symposium: What Ifs and Other Alternative Intellectual Property and Cyberlaw Stories, held in March 2007 and sponsored by the Intellectual Property and Communications Law Program at Michigan State University College of Law.

1. “Don’t Change This License!” (it is copyrighted)	306
2. License Proliferation	308
3. GPLv2 as Applied to the Linux Kernel	310
4. “File-Level” Weak Copyleft	312
5. Dual Licensing and Patent Law’s “Extra Element” Infringement Rule.....	314
6. Microsoft’s Shared Source Licenses	316
CONCLUSION	318

INTRODUCTION

Patent law, by necessity, needs some way to evaluate inventiveness. Otherwise, it will grant rights to advances not worth “the embarrassment of an exclusive patent.”¹ The innovations of version two of the Free Software Foundation’s (FSF) GNU General Public License (GPLv2),² arriving in 1991,³ could not, under U.S. patent law at that time, have been meaningfully measured against patent law’s criteria, often referred to as the five elements of patentability. The first element of patentability, statutory subject matter, would have excluded the GPLv2’s copyright-based licensing technique as a “business method.” While a “process” is statutory subject matter, a “business method” was an exception to the broad potential scope of meaning in the statutory word “process.”⁴ Business methods were not within the domain of patent protection,⁵ so the other four elements of patentability were irrelevant.

1. *Graham v. John Deere Co. of Kan. City*, 383 U.S. 1, 10-11 (1966) (discussing Thomas Jefferson’s worry that “the embarrassment of an exclusive patent” must be correlated to incentives for a non-trivial advance over the prior art).

2. Free Software Found., GNU General Public License, version 2 (1991), <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html> [hereinafter GPLv2].

3. See GLYN MOODY, REBEL CODE: THE INSIDE STORY OF LINUX AND THE OPEN SOURCE REVOLUTION 19, 26-29 (2001).

4. 35 U.S.C. § 101 (2000) (“Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.”).

5. See *State St. Bank & Trust Co. v. Signature Fin. Group, Inc.*, 149 F.3d 1368, 1375-77 (Fed. Cir. 1998) (while acknowledging the prior parlance of an exception to statutory subject matter for methods of doing business, concluding, as to business methods, that it was time “to lay this ill-conceived exception to rest”). The patent at issue in *State Street Bank* claimed a computer system that calculated taxable asset values for a particular configuration of entities sharing participation in pooled mutual funds. *Id.* at 1371; see also *AT&T Corp. v. Excel Commc’ns, Inc.*, 172 F.3d 1352, 1353-54, 1360-61 (Fed. Cir. 1999) (extending the holding of *State Street Bank* to a process claim for a long-distance messaging technique to facilitate charge billing).

Nearly two decades later, at the time of this Article's publication, patent law has changed. The U.S. Patent and Trademark Office (PTO) regularly issues patents for activity that in 1991 would have been ineligible for failing statutory subject matter. Today's issuances go so far as to include patents for methods designed to effectuate a legal outcome.⁶ A copyright-based licensing technique, in 2008, might very well obtain patent issuance if it satisfied the other elements of patentability.

As a path-breaking licensing concept, GPLv2 symbolized a movement this Article will refer to as free and open source software, or "FOSS."⁷ Important strands in the movement depend on ideas that, in combination, were novel to the world of software licensing in 1991, namely requiring generally available public source code disclosure and prohibiting use royalties. Linked to these is the term "copyleft," a pun of copyright and its institutional values, but also a label for a mechanism of reciprocity or extension of FOSS licensing terms, such as source code availability and the anti-royalty provision, to intermixed or further-developed software.⁸ Embodied in a license, these terms implement the FSF's philosophy of functional self-

6. Under statutory subject matter in patent law at the time of this Article, the viability of claims covering human agency to effectuate a legal outcome may depend on how the claim is structured. See *In re Comiskey*, 499 F.3d 1365, 1376-81 (Fed. Cir. 2007). The more the claim covers activity only occurring within human thought, the more it may be characterized as a mental step or process that falls into the "abstract idea" exception to statutory subject matter. *Id.*

7. The FOSS movement has spawned a variety of scholarship in the legal academy. See generally Yochai Benkler, *Coase's Penguin, or, Linux and The Nature of the Firm*, 112 YALE L.J. 369 (2002); David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241, 268, 274 (noting the volunteerism underlying open source software development); Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563. FOSS scholarship also includes an increasing number of books. For an early classic, see OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION (Chris DiBona et al. eds., 1999). See also OPEN SOURCES 2.0: THE CONTINUING EVOLUTION (Chris DiBona et al. eds., 2005); STEVEN WEBER, THE SUCCESS OF OPEN SOURCE (2004). A number of practicing lawyers have authored books on FOSS licensing, and these provide helpful background as well. See, e.g., LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW 103-06, 126-28, 133-136 (2005) (discussing the way in which FOSS licensing developed and how it works).

8. In one sense, "copyleft" expresses the FOSS goal of protecting the general availability of a software work, which is opposite copyright's typical use for software—generally protecting and prohibiting use of the work by others, while perhaps licensing some narrow use for some number of users. In another sense, copyleft refers to a reciprocity rule given in a FOSS license. See ROSEN, *supra* note 7, at 105-06. The FSF, involved in the origination of the label "copyleft," relates it to license term reciprocity with the purpose of software freedom. See Free Software Found., What is Copyleft?, <http://www.gnu.org/copyleft/> (last visited May 14, 2008) ("Copyleft is a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well."); see also Greg R. Vetter, "Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?, 36 RUTGERS L.J. 53, 129-30 (2004) [hereinafter Vetter, *Infectious OSS*] (discussing GPLv2 copyleft).

determination and freedom with one's computer.⁹ The FSF's progenitor, Richard Stallman, implemented these novel licensing concepts in GPLv2 toward his greater end of software freedom. An ingenious yet incomplete document,¹⁰ GPLv2 became the license for important programs generated by Stallman and others through FSF-affiliated software development projects. By its own language, GPLv2 also suggested itself for use on other software.¹¹

A variety of industry developments in the decades following GPLv2's arrival, combined with the license's potent ideological force and clever use of copyright law, propelled FOSS licensing into a prominent and path-breaking place within information technology worldwide. Its force and presence, and lightning-rod character, have grown over time, with GPLv2 remaining the dominant license in mind-share, if not code-share. In addition, all of this occurred without patent protection for GPLv2's unique licensing technique.

This then raises the question: what might have occurred differently if GPLv2's licensing method had been patentable? In other words, if the U.S. patent law of statutory subject matter in 1991 was sufficiently permissive, and if the FSF and Richard Stallman successfully patented the novel licensing approaches of GPLv2, would patent protection have altered the FOSS movement's two-decade trajectory through information technology and the Internet? If so, can we estimate in what ways?

This Article's assessment is that the FOSS trajectory would change minimally, due to a variety of factors, including practical constraints on the enforcement potency of patent claims to GPLv2, competition from other types of FOSS licensing, and strategic considerations for a variety of players and camps within the FOSS movement. However, in the counterfactual, license proliferation diminishes, and dual licensing is foreclosed.

Although no specific legal consequences flow from the mere label, some patents are called pioneer patents because they open a new field or make substantial advances beyond contemporary technology. Would the

9. Free Software Found., The Free Software Definition, <http://www.fsf.org/licensing/essays/free-sw.html> (last visited May 14, 2008).

10. The primary incompleteness of GPLv2 was as much due to the evolution of U.S. patent law as it was due to the license text. GPLv2 mentioned the possibility of patent protection for software covered by the copyright conditions of the license. However, GPLv2 did not explicitly handle granting and terminating permissions to practice software patent rights. This changed in version 3 of the GPL. See Free Software Found., GNU General Public License § 11 (2007), <http://www.gnu.org/licenses/gpl-3.0.html> [hereinafter GPLv3]; GPLv3 FIRST DISCUSSION DRAFT RATIONALE 3-4 (2006), available at <http://gplv3.fsf.org/gpl-rationale-2006-01-16.ps> (discussing the decision to explicitly license patents); see also Greg R. Vetter, *Open Source Licensing and Scattering Opportunism in Software Standards*, 48 B.C. L. REV. 225 (2007) (analogizing the GPL version three revision process to a private standard setting initiative).

11. GPLv2, *supra* note 2, at pmbl. ("You can apply it to your programs, too.")

world think of GPLv2 as a pioneer patent?¹² How would the license be evaluated against the other elements of patentability? How would its inventor, assumed here to be Richard Stallman, use his patent protection, and against whom?

These last two questions structure this Article. Part I starts with patent protection for business methods, not comprehensively, but to differentiate the GPLv2 licensing method from other business method patents. This positioning is important to prepare for Part II, which examines GPLv2's licensing technique against the elements of patentability. The Article concludes, in the counterfactual, that some valid patent claims could have been drafted that are embodied by the software licensing technique of GPLv2.

These claims would have been viable because GPLv2 likely meets the other four elements of patentability under U.S. patent law without changing history.¹³ GPLv2 easily meets the second element of patentability, the requirement for utility. The same is true for the fifth element, the requirement for objective disclosure supporting the claims. The closest questions are with the third element, novelty and the statutory bars, and the fourth element, the requirement for non-obviousness. These are evaluated against the prior art, which would generally include non-secret licensing techniques before Richard Stallman invented GPLv2, which, for discussion, this Article assumes to be in 1991.¹⁴ While a comprehensive evaluation of such "prior art" is not possible in the scope of this Article, it seems likely that GPLv2 was novel in a patent law sense, meaning that it would have satisfied the third element of patentability, assuming no barring events (an assumption necessary to continue the hypothetical).

Obviousness is a closer call. Licensing techniques in use in 1991 may raise obviousness concerns, particularly practices such as sublicensing and grant-back clauses. Without a full prior art review, the obviousness evaluation is truncated to its primary inquiry of asking whether a skilled artisan

12. The parlance of "GPLv2 as a pioneer patent" or "the GPLv2 patent" takes liberties with a more correct statement of counterfactual patent law protection covering GPLv2. However, this Article adopts this loose parlance as necessary throughout the Article for readability. The more precise concept is that the patent right starts with what is recited in a claim near the end of the issued patent instrument. In the hypothetical case of this Article, that claim should cover the technique(s) implemented by the GPLv2 license document, and would be thought of as a "process" claim from a statutory subject matter perspective. The embodiment of this process claim would occur when someone uses the GPLv2 license document by applying it to software. The implications that follow for invalidity based on prior art and infringement are discussed in Parts II and III.

13. While FOSS is a worldwide phenomenon, scope constraints limit the analysis of the counterfactual to U.S. law.

14. This assumption is based on the date of the copyright notice for GPLv2. See GPLv2, *supra* note 2. The date of invention is probably earlier by some small number of years. See MOODY, *supra* note 3, at 26-30. If so, that is not critical to the Article's analysis since it does not seek to precisely define the dates establishing prior art and barring events.

would find the differences between the claimed invention and the prior art to be obvious.¹⁵ However, obviousness has a secondary inquiry, which looks at considerations external to the claimed functionality. These include such notions as “teaching away” from what is claimed, unexpected results, and “long felt” need.¹⁶ The success of GPLv2 within the remarkably successful FOSS movement indicates that the secondary considerations point to a finding of non-obviousness.

Since the third and fourth elements of patentability are particularly sensitive to claim language and scope, the Article will present one sample hypothetical claim while recognizing that this is both an overly narrow and overly broad approach. It is narrow because basing the analysis on one sample claim ignores the typical patent drafting technique of fashioning a variety of claims varying in scope.¹⁷ It is overly broad because the claim scope has not been fashioned against a rigorous prior art evaluation, and thus might be too broad to be valid. The sample claim is undoubtedly imperfect, and others may very well see ways to improve its coverage of GPLv2.

Throughout the analysis of Part II, one goal is a rough assessment of the likelihood of obtaining patent protection for some claims of non-trivial scope. Another goal, however, is to note the perspective cast on FOSS licensing by patent law’s criteria for measuring inventiveness. A great challenge in patent law is to make these criteria sufficiently objective to allow evaluation by the courts and the PTO. While imperfect, patent law’s elements of patentability provide such a framework. The Article argues, in essence, that GPLv2 floats through these criteria with only slight worries about the obviousness inquiry. If correct, this is a testament to GPLv2’s inventiveness and its contribution to FOSS, software licensing, and information technology.¹⁸

The counterfactual possibilities for the second question are less bounded than the first. Estimating what parties would do if they had a patent that covers an activity or technology in the real world may seem easy—sue infringers, or at least threaten suit and try to license. A related choice might be whether to enjoin the infringer and keep the market closed (often

15. 35 U.S.C. § 103 (2000).

16. *Graham v. John Deere Co. of Kan. City*, 383 U.S. 1, 17-18 (1966).

17. Phillip M. Pippenger, *Prosecuting Patents with an Eye toward Enforcement*, 910 PLI/PAT 1239, 1239, 1245 (2007) (discussing the need for and methods to achieve a “carefully tailored claim set” anchored in both process and product statutory subject matter).

18. While a variety of instrumental or incentive-based theories have been advanced to support the patent system, one theory in particular emphasizes the role an issued patent might have to reflect positively on a technology. See Clarisa Long, *Patent Signals*, 69 U. CHI. L. REV. 625, 635, 646-49 (2002) (proposing that facilitating the option to credibly publish information is an important role for the patent system).

the answer for a competitor that infringes) or license the infringer regardless.

These choices apply to only a small percentage of patents because most patent claims do not cover anything in the real world. Thus, for commercialization purposes, these patents have no value.¹⁹ For prior art purposes, they have value, but not in a direct wealth-generating sense. The U.S. patent system expressly allows filing a document that states what a patent discloses, called a statutory invention registration (SIR).²⁰ The SIR has no claims, but its disclosure places technology into the prior art.²¹ This ensures that no one in the future can obtain patent protection for information the SIR discloses without further developing the technology. This type of publication value also occurs with general publication or dissemination of knowledge or technology. GPLv2 has sufficiently wide dissemination to count as prior art from sometime in the early 1990s, so this part of the story is not counterfactual, but actual.

Thus, the analysis, by assuming a hypothetical GPLv2 patent, also assumes that infringing real-world activity occurs, since a great number of FOSS programs have applied the GPLv2. In addition, many other FOSS-form licenses have imitated the GPLv2, any deployment of which might also be infringing activity.

Given the assumption of a patent for such software licensing with potent claims, what would Richard Stallman and/or the FSF have done with patent protection for GPLv2? Such a question involves the dynamics of the entire information technology industry, all of which the FOSS movement has impacted.²² It includes contrasting ideologies symbolized by Richard Stallman and the FSF, as compared to other groups willing to entangle FOSS with commercial interests to a greater degree. FOSS licensing can make strange bedfellows and has gathered corporate advocates as well known as IBM, even though at first glance, the FOSS premise of open, shareable source code is in opposition to the type of traditional software licensing approaches IBM championed in earlier decades.

19. Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495, 1497, 1508, 1531-32 (2001); see generally John R. Allison et al., *Valuable Patents*, 92 GEO. L.J. 435, 437 (2004) (arguing, among other points, “that the easiest way to discover the characteristics of valuable patents is to study litigated patents”).

20. 35 U.S.C. § 157.

21. See USPTO, 1111 SIR Publication and Effect—1100 Statutory Invention Registration (SIR) and Pre-Grant Publication (PG Pub), available at http://www.uspto.gov/web/offices/pac/mpep/documents/1100_1111.htm (last visited May 14, 2008) (“[A] published SIR will be treated the same as a U.S. patent for all defensive purposes The waiver of patent rights to the subject matter claimed in a statutory invention registration takes effect on publication (37 CFR 1.293(c)) . . .”).

22. See Kim Polese, *Foreword: Source is Everything*, in OPEN SOURCES 2.0, *supra* note 7, at x.

The counterfactual proceeds with Richard Stallman and/or the FSF as the patent owner, and changes only one major historical variable: subject matter eligibility for patent protection. The analysis would be dramatically different, and of greater length, if it changed a second variable and put ownership of the claimed method elsewhere, such as with Microsoft. Recognizing this, the analysis may occasionally refer to the implications of hypothetical alternative owners.

Patents are often enforced against a variety of targets. Thus, to organize Part III, and remove from pure speculation what one person or group might do, this Article proceeds by cataloging some of the more prominent events or situations with structural or institutional consequences for the FOSS movement. These are cases where FOSS licensing approaches might be subject to suit under the hypothetical GPLv2 patent. Examining each will provide a sense of the possibilities, considerations, and constraints channeling enforcement at each moment.

The Article concludes by emphasizing the themes above, and noting that patent protection for FOSS licensing techniques is more than a hypothetical in 2008. While they do not contain pioneering concepts of broad scope like the GPLv2, a small but growing number of patent applications are simmering at the PTO and claim licensing techniques arising from the FOSS movement.²³ What if these patents issue? How will they be enforced in the future? The FOSS movement likely will not wait two decades for answers to these questions.

I. BUSINESS METHOD PATENTS AT THE INTERSECTION OF SOFTWARE AND LICENSING

When the United States Court of Appeals for the Federal Circuit eliminated the “business methods” exception to the domain of patents, it declared the exception “ill-conceived.”²⁴ Categorizing statutory subject matter leads to metaphysical debates of an intractable nature.²⁵ In the period covered by the counterfactual, the primary controversies in statutory subject

23. See, e.g., *Method and Apparatuses for Reviewing General Public Licenses*, U.S. Patent Pub. 2006-0288421 (filed June 15, 2005) (disclosing a method and apparatuses for detecting conflicts in licensing terms between a first general public license corresponding to a first group of code and a second general public license corresponding to a second group of code). See *infra* note 125.

24. *State St. Bank & Trust Co. v. Signature Fin. Group, Inc.*, 149 F.3d 1368, 1375-77 (Fed. Cir. 1998).

25. *In re Nuijten*, 500 F.3d 1346, 1367 (Fed. Cir. 2007) (“[T]he outer limits of statutory subject matter should not depend on metaphysical distinctions such as those between hardware and software or matter and energy, but rather with the requirements of the patent statute: is an invention a ‘process,’ ‘machine,’ ‘manufacture,’ or ‘composition of matter’ . . .”).

matter arose from biotechnology and software. Inventions in each technology are claimable as either processes or structures. Importantly, neither is typically so abstract²⁶ as to operate mostly within human thought. Software, though abstract, can be easily claimed in relation to computing structure. In fact, being patented as a “structure” was software’s first avenue into statutory subject matter, the second being its inherent correspondence with the statutory word “process.”²⁷ As business increasingly computerized its processes using information technology, the historical “business methods” exception fell before this wave of automation and patenting related to the automating technology.²⁸

The approach of this counterfactual will shortly render the subject moot because it will stipulate the first element of patentability to facilitate the analysis. Before that departure, however, there is a need to differentiate the availability of patent protection for computing methods from protection for activity such as FOSS licensing.

Patent protection for software, as opposed to licensing methods, worries the FOSS movement, but these concerns are not this Article’s focus, except to note an area of overlap—license enforcement is sometimes automated using various technologies, some of which might be categorized under the broad rubric of digital rights management. Consider this “software patent” example: claiming an innovative method to sort information in a particular technological context.²⁹ Such a patent’s claims will be written with varying degrees of dependence on underlying computing structure. Alternatively, a patent covering, for example, a method of arbitration, may have some claims that describe the method without reference to any structure.³⁰ Such claims recite the steps to a process that activates and imple-

26. *Diamond v. Chakrabarty*, 447 U.S. 303, 309 (1980) (noting that “[t]he laws of nature, physical phenomena, and abstract ideas have been held not patentable” (citing *Parker v. Flook*, 437 U.S. 584 (1978))).

27. Dan L. Burk, *The Problem of Process in Biotechnology*, 43 HOUS. L. REV. 561, 564, 589-90 (2006) (noting that “[w]hether contemplating patents in practice or in theory, attorneys and scholars typically divide the universe of patents into *product* patents, which encompass the categories of machines, compositions, and manufactures; and *process* patents, which cover the odd category out—that of new and useful processes” toward observing that software patents, even more directly than biotechnology structures, correspond to direct informational patenting).

28. In the pre-computer era, business methods were not related to software. A classic example is a method for accounting, or a technique to demonstrate a product. Only later, when software began to underlay more business operations did patent law for business methods and for software begin to merge and influence one another. *See generally State St. Bank & Trust Co.*, 149 F.3d at 1375; *AT&T Corp. v. Excel Commc’ns, Inc.*, 172 F.3d 1352, 1353-54, 1360-61 (Fed. Cir. 1999).

29. *See* U.S. Patent No. 7,103,603 (filed Mar. 28, 2003) (entitled “Method, apparatus, and system for improved duplicate record processing in a sort utility”).

30. *In re Comiskey*, 499 F.3d 1365, 1379-81 (Fed. Cir. 2007).

ments thought (and perhaps corresponding action) among a group of humans. This approach may be too abstract to meet the statutory subject matter element because the steps are all “mental”—they occur in human thought.³¹ The potential overlap of patenting for a licensing method with software is when software is used to enforce a licensing approach. Claims can be mostly about the licensing terms, but can include elements/limitations of computing structure that will enforce, present, or relate to the terms. For statutory subject matter, this will likely validate the claims for the first element of patentability.³²

Thus, if the copyleft licensing techniques of GPLv2 were conceived in 2008, patent law would readily allow the crafting of some claims that meet the statutory subject matter. These claims would be potent in that they would minimally enmesh with computing structure, and thereby avoid the mental steps notion of an excluded abstract idea. With the increasing involvement of software to handle licensing terms, much activity might fall within the patent’s enforcement power. This could include not only complex technologies, such as digital rights management for license enforcement, but also simpler involvement, such as the increasing propensity of software to present licensing terms to the user/installer and to take and store an indication of assent.

To proceed as simply as possible, this Article will model the GPLv2 patent claims near the edge of what would be considered statutory subject matter at the time of the Article’s publication. The one sample claim given below is not encumbered with artificially inserted structure in order to meet the statutory subject matter. By the magic of the counterfactual, that first element of patentability will be stipulated.

II. CLAIMING COPYLEFT: FOSS LICENSING UNDER GPLv2 AS A PATENT PROTECTED ACTIVITY

Copyleft, as a reciprocity mechanism, must propagate terms through successive revisions of a program in order to facilitate its continued trans-

31. Patent law has a subject matter exclusion that can be characterized as a specific type of the general “abstract idea” exception to statutory subject matter. This is the “mental steps” doctrine, where a claim reciting only process steps occurring mostly within human thought are not eligible. *Id.* at 1377-78.

32. In many situations, entangling an abstract idea such as a mental process with structure will allow the claimed method to meet the statutory subject matter requirement.

When an unpatentable mental process is combined with a machine, the combination may produce patentable subject matter While the mere use of the machine to collect data necessary for application of the mental process may not make the claim patentable subject matter . . . these claims in combining the use of machines with a mental process, claim patentable subject matter.

Id. at 1379.

parency and sharing.³³ The FSF provides a description of the process as follows:

To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code or any program derived from it but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable.³⁴

In the GPLv2, the distribution terms require that source code be available and prohibit use royalties.³⁵ In patent law terminology, several items³⁶ come together in cooperative combination to achieve a new end. A variety of other topics in GPLv2 will not be the focus in this Article or, more importantly, with the sample claim.³⁷ The items covered by the GPLv2 patent's claim(s) are central to FOSS's novel approach to software licensing and its resulting freedoms and development advantages. In other words, the claims, particularly claim 1, should cover the heart of GPLv2.³⁸

A. A Hypothetical Claim for the GPLv2 Patent

Below is a sample approach to illustrate the theoretical content of a patent claim embodied by GPLv2. This is written as claim 1, which patent law envisions as the broadest claim among the several claims that typically issue in a patent.

33. See McGowan, *supra* note 7, at 258-59 (describing the reciprocity of copyleft as a "web of blocking copyrights").

34. Free Software Found., What is Copyleft?, *supra* note 8.

35. GPLv2, *supra* note 2, §§ 2(b), 3.

36. By using the word "items" here, the Article is avoiding a semantic issue in patent law about what term to use to describe a subset of the claim: element or limitation. Suffice to say, either term conveys that some of the claim language describes one part of the items that come together in the cooperative combination that is the patent claim.

37. Items from GPLv2 not included in the claim include, among others, provisions prohibiting discriminatory licensing, provisions disclaiming warranties and indemnification, and provisions dealing with certain technical details about making source code available. See GPLv2, *supra* note 2 (listing sections of the license not involved in the hypothetical claim).

38. GPLv2 has been imitated in a variety of ways. One example of these replicas is an innovative effort to simplify the GPLv2, called the SimPL. Robert Gomulkiewicz, Simple Public License (SimPL), <http://www.law.washington.edu/CASRIP/License> (last visited May 14, 2008) ("The SimPL is intended to demonstrate a simpler approach to the GPL that achieves the same goals. Above all else, it is intended to be read and understood by the average open source developer, who may have only a limited understanding of the words of the GPL."). Earlier versions of the claim given in this paper, as well as the example claim used to present the analysis of this paper at the symposium generating this Article, were based on SimPL as a very helpful first step toward capturing the essence of GPLv2 in the format of a patent claim.

1. A method of licensing software from a licensor to at least one licensee, comprising:
- (a) issuing software by the licensor in source or object code form for any licensee to undertake activity with the software;
 - (b) the activity comprising at least one item of any of: (i) non-modifying activity, which comprises copying, running, or using the software; (ii) modifying activity, which comprises modifying or making derivative works of the software wherein incorporated programming comprises intermixed software; or (iii) distributing activity with or without any other activity;
 - (c) the activity not being conditioned on usage royalty payments from any licensee to licensor;
 - (d) conditioning the software issuance such that any licensee activity with the software must comply with license terms; and
 - (e) license terms comprising: (i) a non-distribution permission set of conditions, wherein if the licensee does not distribute the software, the activity is non-restricted; and (ii) a distribution permission set of conditions, wherein upon issuance of the software by the licensee's distribution of the software:
 - (A) the licensee must make source code available to its subsequent licensees;
 - (B) the licensee must not condition its issuance of the software on receipt of royalty payments from any subsequent licensee for activity with the software;
 - (C) the licensee must condition the subsequent licensee's activity on the license terms; and
 - (D) the licensee must license intermixed software under the license terms when the licensee engages in modifying activity.

As mentioned above, the sample claim is undoubtedly imperfect, and others may very well see ways to improve its coverage of GPLv2, or see alternative drafting approaches to maximize the scope, and, thus, maximize infringement coverage, of the claim, while minimizing invalidity problems.³⁹

Having presented the sample claim and stipulated to the first element of patentability, Section II.B discusses patent law's requirement that the claimed invention have utility.⁴⁰ Section II.C discusses the requirement that the patent application provide sufficient objective disclosure to support the

39. As the first claim in the hypothetical patent, the sample claim consciously seeks to recite a minimal number of elements/limitations. Below are samples of possible dependent claims:

2. The method of claim 1, wherein the license terms further comprise that licensee must leave in place any copyright or other intellectual property notices in the software source code or in its associated files.

3. The method of claim 1, wherein the distribution permission set license terms further comprise that licensee must document any changes made to the source code of the software by annotations in the source code of the software.

40. 35 U.S.C. § 101 (2000).

claims.⁴¹ After these two easily met requirements are satisfied, the remaining sections examine the prior-art-based elements of patentability.

B. Utility

The GPLv2 patent easily meets the utility requirement. United States patent law authorizes the PTO to issue patents for “useful” inventions. Like the first criterion, statutory subject matter, the utility criterion has diminished as a barrier to patent protection. A *de minimis* practical utility will suffice. The claimed technology must provide some practical, identifiable benefit. The utility need not meet any particular moral or other standard, but the claimed invention must do something to accomplish its described purpose.⁴²

A counterexample illustrates the utility requirement. The PTO regularly rejects patent applications for perpetual motion machines due to lack of utility. The PTO rejects these applications because recognized science disproves the possibility of usefulness for these devices.⁴³

None of these concerns affects the GPLv2 patent. In 1991, there was recognition of the usefulness of source code availability for interoperability and technical transparency. Sharing software also had a place in the software ecosystem of that era, although the sharing mechanism was different.⁴⁴ These software licensing models did not envision sharing as collaborative software development like that of the FOSS movement, but such precursor practices were clearly useful under patent law’s low utility threshold.

Additionally, with the benefit of nearly twenty years of actual history, FOSS licensing clearly meets patent law’s conception of utility. Its approach continues to challenge conventional paradigms about software de-

41. *Id.* § 112.

42. ROGER E. SCHECHTER & JOHN R. THOMAS, INTELLECTUAL PROPERTY: THE LAW OF COPYRIGHTS, PATENTS AND TRADEMARKS 315 (2003).

43. *Id.* at 318.

44. Before the FOSS movement rose to prominence, there was an active trade in freeware and shareware software. *See, e.g.*, INT’L INST. OF INFONOMICS, UNIV. OF MAASTRICHT & BERLECON RESEARCH GMBH, FREE/LIBRE OPEN SOURCE SOFTWARE: SURVEY AND STUDY: PART III—BASICS OF OPEN SOURCE SOFTWARE MARKETS AND BUSINESS MODELS 11-12 (2002), available at http://www.berlecon.de/studien/downloads/200207FLOSS_Basics.pdf, report overview page, available at <http://www.infonomics.nl/FLOSS/report/> [hereinafter FLOSS Basics] (presenting a two-by-two classification scheme for software using axes of source code availability and royalties, resulting in the following four categories: shareware/freeware, commercial open source software, noncommercial open source software, and proprietary/commercial software). These programs were usually developed and written in a similar vein as single-programmer FOSS projects. Hobbyists and tinkering programmers built interesting or useful software and then offered it, typically in object code form, to others free of charge, or in the case of shareware, for a *de minimis* fee. Compared to the FOSS movement, what was missing was the distributed collaborative development model enabled by FOSS licensing.

velopment methodologies and sources of innovation in the information technology industry. The software resulting from this movement occupies an important place within the industry's ecology, particularly within the Internet. In fact, much of the Internet's enabling software came about in conjunction with the FOSS movement. The movement successfully scaled with the Internet's growth due to its source code transparency and shareable nature, both being features of its licensing approach.

C. Objective Disclosure

The objective disclosure element of patentability is completely within the control of the patent applicant since she prepares the disclosure documents that must accompany the original patent application. As a result, this analysis will stipulate that the disclosure requirement has been met.

As a legal document, an issued patent has differing sections, including the claims and supporting information,⁴⁵ all of which together form the document's disclosure. One purpose of this information is to populate the public domain when the patent expires. Another is to enable artisans to make and use the invention. As such, the disclosure requirement has multiple sub-tests, one of which is "enablement."⁴⁶ Functioning alongside a related requirement, the "written description" test, enablement asks whether the patent document gives a person of ordinary skill in the technology enough information to make and use the claimed invention without undue experimentation.⁴⁷ Though there are two other sub-tests, they are of lesser import for this Article's purposes.⁴⁸ In any case, if a claim fails to meet any one of the sub-tests, it is invalid.

45. The preceding pages before the claims, typically a handful to several dozen, support the patent's exclusory right and allow society to benefit from the technology disclosed in the patent. When the patent expires, it leaves a document that gives no one a right to exclude, but which may be significant for its technical content. SCHECHTER & THOMAS, *supra* note 42, at 393-95. The U.S. patent system has over seven million issued patents at the time of this writing. See United States Patent and Trademark Office, Table of Issue Years and Patent Numbers, for Selected Document Types Issued Since 1836, <http://www.uspto.gov/web/offices/ac/ido/oeip/taf/issuyear.htm> (last visited May 14, 2008) (displaying that utility patent numbers in 2008 are well above the seven million threshold). This creates a vast database of technical information even though many of these issued patents are expired. For this trove to be useful, each patent has to provide sufficient information to allow artisans to practice the claimed technology.

46. SCHECHTER & THOMAS, *supra* note 42, at 394-98.

47. *Id.*

48. The other two sub-tests under the disclosure requirement are "best mode" and "definiteness." With best mode, the patentee must disclose her best way of practicing the claimed technology, if she had one. The definiteness requirement applies to the claims—they must be sufficiently specific to provide some sense of the boundary of the invention.

In order for the GPLv2 patent to meet the objective disclosure requirement, this requires a sufficient explanation to a software licensing professional of that era of how to implement and practice the claimed method of licensing, and requires specifying if there were any preferred or best modes of implementing its claimed method. This is inherently achievable, although the GPLv2 license document does not necessarily accomplish this task alone. Assume an experienced patent attorney prepares the application. As input, she will start with the GPLv2 license and any other explanatory documents used in its development to prepare the disclosure of the original application. The typical approach may also include interviewing the inventor in addition to reviewing the inventor's documents and records. For a commercially valuable patent, the originally-filed disclosure will likely be faced with decades of scrutiny, so the incentives to provide sufficient information are high.

D. Novelty and Statutory Bars

Utility is a low threshold, and disclosure is under the applicant's control, but the novelty requirement brings with it the possibility of surprise. Novelty asks whether a single prior art reference has or discloses what the claim recites. If so, patent law says that the prior art reference renders the claim invalid by anticipating the claim. The word "anticipate" in this construct is a term of art, meaning that the reference has, in itself, all of the elements/limitations of the claim.⁴⁹

Statutory bars relate to novelty in that a reference must also anticipate the claim in order to render the claim invalid. Below this high-level description, there are a host of details distinguishing novelty from statutory bars under U.S. law, many of which relate to timing questions about what dates count for what purposes.⁵⁰ These details will not make a difference here, since the analysis necessarily takes a general approach.

49. SCHECHTER & THOMAS, *supra* note 42, at 364-65.

50. U.S. patent law measures novelty from the date of invention, but measures statutory bars from a date keyed to the applicant's filing date (although what that date is may also require a legal inquiry, depending on the procedural course of the patent prosecution). *Id.* at 323-26. The patent statute says that even if no one else has developed the claimed technology before an inventor, the inventor may be prohibited (barred) from using the patent system to protect the technology if the inventor commercializes the technology in particular ways more than one year before filing for a patent. 35 U.S.C. § 102(b) (2000). The most common barring activities include: (i) using the claimed technology for commercial ends; (ii) selling or offering to sell the technology; and (iii) publishing the technology. The statutory bars support two important policies for patent law: promoting prompt disclosure of newly discovered technology and limiting the period of control over patented technology. SCHECHTER & THOMAS, *supra* note 42, at 325-27.

The novelty requirement may bring surprise because the applicant and the PTO may not discover all prior art at the time of the application. Prior art discovered years after the patent issues, and which was not uncovered at the time of the application, might help invalidate the claimed invention. In fact, due to the intense search efforts during patent litigation, prior art is often discovered that was not before the PTO.

This Article will not evaluate the prior art using a search as might guide the PTO or a court in litigation. Even without such a prior art search, however, this Article posits a high likelihood that the GPLv2 patent would have some claims not anticipated by the prior art. This is in part an anecdotal estimate based on the author's involvement with computing beginning before GPLv2. In part it is also hindsight—as the FOSS movement has grown, it has attracted increasing scrutiny and scholarship, which has yet to unearth clearly anticipating licensing methods. The estimate also relies on the nature of patent claims—adding more elements/limitations decreases the probability that a prior art reference anticipates. For example, the two dependent claims shown in footnote 39 are each harder to anticipate than sample claim 1 because they are more specific, i.e., they add more elements/limitations.

E. Non-Obviousness

The gist of the non-obviousness requirement comes from the plain meaning of the word “obvious:” do not grant the patent right for claimed technology that is obvious.⁵¹ In other words, non-obviousness requires a technical advance greater than a merely trivial or minor improvement. Evaluating obviousness, however, is easier said than done. In order to assist the inquiry, patent law compares the claim to the prior art.⁵² The comparison is similar to the anticipation test for novelty, but multiple prior art references can work together to invalidate the claim under obviousness.⁵³ In novelty and the statutory bars, a single prior art reference must disclose what the claim recites. In other words, the novelty inquiry frames the claim as a composite of cooperative elements and determines if any single refer-

51. SCHECHTER & THOMAS, *supra* note 42, at 369-71.

52. For a reference to be “prior art,” it must fall into one of the statutorily-defined categories, and be “prior”—a timing and date-pegging question that itself may need legal inquiry to answer. *Id.* at 371-73. Moreover, the prior art must be “analogous,” one meaning of which is that it is from the “field” of the invention. *Id.* For the GPLv2 patent, that field will be deemed to be the practice of software licensing. To see how this might impact the analysis, if the “field” were expanded to include all information licensing, then all music or literary work licenses would be included. This vastly increases the potential universe of prior art, increasing the chances of finding more prior art references that can support the obviousness argument to invalidate the claim.

53. JANICE M. MUELLER, AN INTRODUCTION TO PATENT LAW 189-90 (2d ed. 2006).

ence has all the elements. In evaluating obviousness, on the other hand, multiple prior art references can supply the elements. Even if no single reference combines the elements as claimed, the claim might still be invalid if an artisan in the technology would find it obvious to make the claimed combination.

Improper hindsight occurs when the post-invention knowledge of the claimed technology's workings causes the inquiry to veer too readily toward a finding of obviousness.⁵⁴ Objective criteria to prevent hindsight have been a challenge to patent law, and likely always will be, due to the inherent nature of the hindsight phenomenon. Post-invention knowledge of a different sort, however, can also influence the obviousness inquiry. Namely, the so-called "secondary considerations," which look beyond the artisan's view of the differences between the prior art and what is claimed, can account for effects of the technology after the invention and conceptions about what was invented before the invention.⁵⁵

1. Differences Compared to the Prior Art of Software Licensing

This Article's analysis estimates that most software licensing professionals with extensive experience before GPLv2 would count all of the elements/limitations of the GPLv2 patent's claims as present, or mostly present, in the prior art of software licensing. Source code licensing existed, but sometimes with secrecy conditions. Licensing of both source and object code in a publicly or generally available manner also existed, both with and without secret source code, and with and without conditions on use and/or redistribution. The license agreements of the time existed to control royalty rates, and some of these would specify that software be distributed without charging royalties. These practices all operated at a time when the dominant software licensing model consisted of secret-source code, and charging royalties for proprietary software.

Subsequent licensing was, and is, common under the label "sublicensing." The label covers a variety of practices within proprietary licensing, each of which may be specifically or broadly defined. For example, often-times a distributing licensee receives a right to sublicense to other entities, but the sublicense occurs under a fully or partially specified end-user license agreement (EULA). Other methods eschew this approach and specify broad provisions that the sublicense must meet, leaving the implementation details

54. On the hindsight bias effect on patent law's obviousness determination, see Gregory Mandel, *Patently Non-Obvious II: Experimental Study on the Hindsight Issue Before the Supreme Court in KSR v. Teleflex*, 9 YALE J.L. & TECH. 1 (2007), and Gregory N. Mandel, *Patently Non-Obvious: Empirical Demonstration that the Hindsight Bias Renders Patent Decisions Irrational*, 67 OHIO ST. L.J. 1391 (2006).

55. See MUELLER, *supra* note 53, at 186-92.

up to the sub-licensor (who is also the licensee). Different royalty rate models also add to the diversity. Like FOSS licensing generally, sublicensing envisions a chain of distribution. One difference, however, is that the FOSS approach allows the chain to self-propagate many more links than the typical three-level licensor-to-licensee-to-sublicense model of proprietary software. Moreover, the distribution patterns in FOSS can morph into a multi-dimensional web structure, although practical constraints limit this phenomenon.⁵⁶

Beyond sublicensing, licenses have provisions to handle who took rights for modifications, and what a licensee can do with modifications to software. These can include reciprocity provisions requiring those with licensing interests in the code to “grant back” rights to others. For this reason, such provisions are often called grant-back clauses.

However, even if all the elements/limitations recited in the claim existed in the prior art, that by itself does not prove obviousness. The artisan in the field, patent law’s “reasonable person,” must have thought it obvious to make the claimed combination. In litigating this issue, expert testimony is typically offered. Courts, however, often find the expert evidence in equipoise. For that reason and others, the obviousness inquiry also uses “secondary considerations.”

2. *Secondary Considerations: Particularly “Unexpected Results”*

Some secondary considerations take into account perspectives about the claim’s technological field before the invention, while others take into account what occurs after the invention and its subsequent commercialization, if any. Among the secondary considerations,⁵⁷ this analysis chooses to focus on just one: unexpected results. Specifically, this analysis will focus on the unexpected success of the licensing model, both in terms of propagation of FOSS licensing and the development of software under the model.

56. Most FOSS project distribution patterns are not an ad hoc web because there is semi-centralization of the code base via web repositories in association with a group of project leaders who disproportionately influence development, due to their early history with the project or other unique attributes. A related phenomenon is forking, where a group of programmers takes a FOSS project in a different direction, effectively creating two competing (or complimentary) products where before there was only one. FOSS licenses allow forking, as long as the software continues to meet the conditions of the original license regime. Forking is rare, but the possibility of forking is thought to provide an important disciplining force on the ad hoc leadership group in charge of most open source products. McGowan, *supra* note 7, at 263-64.

57. Secondary considerations include commercial success, unexpected results, copying, long-felt but unresolved need, and the failure of others to develop the invention. *Graham v. John Deere Co. of Kan. City*, 383 U.S. 1, 17-18 (1966).

FOSS licensing propagates in two ways. First, the GPLv2 has numerous imitators. These other licenses flatter the GPLv2 model of shareable, transparent software that must remain in that cycle. Second, an increasing amount of code in use is under the GPLv2 or its imitators. The latter point, however, needs greater empirical support for full potency. For example, while more and more code is indisputably under GPLv2-style licenses, is this increase greater than other growth measures in information technology, such as the installed base of equipment or the total amount of all software installed? Is it more than the increase in attribution-only FOSS code? While important, this additional quantification is not critical. Rather, the crucial notion is this author's estimate that, before 1991, most software licensing professionals would scoff at the proposition that the GPLv2 scheme would have more than niche success. Therefore, any non-trivial FOSS licensing presence is significant, and the FOSS movement is clearly more than niche.

Related to this is the unexpected success of the software development model enabled by FOSS licensing. It is becoming increasingly difficult to stereotype FOSS development. However, for the most prominent projects, there are some common elements. One oft-celebrated feature is FOSS development's distributed nature, engendered by source code transparency. The programmers are scattered, possibly around the globe, and use the Internet to coordinate activities with the shareable code base. Volunteerism, or at least subsidization, fuels the projects. Either or both could come from an individual or organization.

FOSS's volunteer, distributed development model is also unique in allowing programmers to self-select work within the project.⁵⁸ One can earn one's way onto a desired part of the project by contributing superior code for that part. The model also has software defect detection advantages. While programming is as much art as science, good technological solutions are recognizable. These inner workings are observable to the user community, which can also contribute, although in different ways than can the core programmers.

The FOSS development model has influenced the world of commercial software development, lessening its traditional command and control approach,⁵⁹ and prompting companies to reevaluate internal code sharing

58. See Benkler, *supra* note 7, at 414-15 (noting that an advantage of open source software and peer production is that, compared to management hierarchies, contributors are better able to judge where best to apply their talents within various projects).

59. For the classic treatment of the differences in traditional proprietary software development as compared to FOSS, see Eric S. Raymond, *The Cathedral and the Bazaar* (2000), <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar> (coining the phrase "[g]iven enough eyeballs, all bugs are shallow," a metaphor about eliminating defects in software with massively paralleled human examination of the problem).

practices and collaborative structures. These effects would not be expected by a software licensing professional examining the GPLv2 for the first time in 1991.

This treatment of “unexpected results” as a secondary consideration overlaps with two of the other considerations: copying of the GPLv2 licensing method and the commercial success based upon it. The latter notion takes the analysis away from the question of patent claim validity and instead places the focus on the question of enforcement. The FOSS movement is involved with commercial information technology interests in many ways. Thus, even for FOSS programs whose licensing disables traditional royalty payments, there can be, and is, commercial success.⁶⁰ Commercial success means activity in commerce and in the information technology ecosystem. For a patent with a method claim, such activity may constitute a “use” of the method, and thereby potentially be considered an act of infringement.⁶¹

III. CLAIMED COPYLEFT: ENFORCEMENT MOMENTS IN THE FOSS MOVEMENT

Like many movements, as its success surged, the FOSS movement became increasingly multi-stranded. There are many ways to taxonomize the interests involved with FOSS, but to simplify the discussion below, the Article will use a two-camp approach: the free software camp and the open source camp.⁶² The open source camp emphasizes the software development advantages arising from FOSS licensing, and contains Linus Torvalds, the leader of the Linux kernel project. The free software camp emphasizes self-determination possibilities and social solidarity arising from FOSS licensing, and contains the FSF and the putative GPLv2 patent holder, Richard Stallman. The Linux kernel project is the basis for a number of operating system distributions that are popularly called “Linux,” although the FSF argues that “Linux” is more appropriately called “GNU/Linux,” emphasizing the principles of software freedom associated with the GNU project. In general, a GNU/Linux operating system distribution rests on the Linux ker-

60. For example, Red Hat distributes an operating system based on the Linux kernel, which is a GPLv2 application. See generally Red Hat, The Open Source Leader, <http://www.redhat.com/about/> (last visited May 14, 2008) (“Today Red Hat is the world’s most trusted provider of Linux and open source technology.”).

61. The patent owner has legal power, through an infringement lawsuit, to stop others from making, using, selling, offering to sell, or importing whatever the patent validly claims. 35 U.S.C. § 271(a) (2000).

62. See Greg R. Vetter, *Exit and Voice in Free and Open Source Software Licensing: Moderating the Rein over Software Users*, 85 OR. L. REV. 183, 205 (2006) [hereinafter Vetter, *Exit and Voice in FOSS*] (noting that the line between the two camps is not a bright line).

nel, but contains critical components from the GNU project.⁶³ The FSF's vocabulary control argument is but one example of the group's explicitly political orientation and its proclivity to evangelize the merits of free software.⁶⁴

A. The FSF and Copyright Enforcement of GPLv2

Evangelization is also in the GPLv2 license itself, both expressively and functionally. The license's one-page preamble is of a constitutional character for the free software camp.⁶⁵ The terms implementing its novel software licensing approach seek to condition distribution of software in such a way as to ensure that it remains freely reviewable and shareable through a web of distribution and generational future development. The conditions that prohibit privatizing the code activate based on distribution, in the copyright sense. When such privatizing occurs, the FSF usually enforces the GPLv2 through conversation with the offending entity, but it has used court action as well.⁶⁶ Both enforcement modes help emphasize the

63. The GNU/Linux operating system is sometimes referred to as Linux. An operating system, however, is not a single large software work, but is rather an aggregation of many software components. The central component is the kernel, which is properly called Linux. Distributions of a Linux kernel-based operating system include other critical components. Most distributions include a set of essential software tools from the GNU project, a separate open source software effort. Richard Stallman, *The GNU Project*, available at <http://www.gnu.org/gnu/thegnuproject.html> (found under the heading "Linux and GNU/Linux"). Thus, some use the name GNU/Linux for such a distribution. *Id.* ("We call this system version GNU/Linux, to express its composition as a combination of the GNU system with Linux as the kernel."). The GNU acronym is a self-referential label meaning "GNU's Not UNIX," with Unix being a predecessor computer operating system. See *The GNU Operating System*, <http://www.gnu.org> (last visited May 14, 2008).

64. See Richard Stallman, *Why "Open Source" Misses the Point of Free Software*, <http://www.gnu.org/philosophy/open-source-misses-the-point.html>.

65. Rod Dixon, *Breaking into Locked Rooms to Access Computer Source Code: Does the DMCA Violate a Constitutional Mandate When Technological Barriers of Access are Applied to Software?*, 8 VA. J.L. & TECH. 2, ¶ 106 n.257 (2003); LI-CHENG (ANDY) TAI, *THE HISTORY OF THE GPL* (2001), http://www.free-soft.org/gpl_history/; Free Software Found., *GPL Version 3: Background to Adoption*, June 3, 2005, <http://www.fsf.org/news/gpl3.html>.

66. *Progress Software Corp. v. MySQL AB*, 195 F. Supp. 2d 328 (D. Mass. 2002); see also Counterclaim at 5, *Progress Software Corp.*, 195 F. Supp. 2d 328 (No. Civ. A. 01-11031-PBS) (alleging that Progress Software, through its wholly-owned subsidiary NuSphere, violated the GPL by "selling derivative works based on the MySQL™ Program without making the underlying source code available"); cf. Declaration of Eben Moglen in Support of Defendant's Motion for a Preliminary Injunction on its Counterclaims at 11, *Progress Software Corp.*, 195 F. Supp. 2d 328 (No. Civ. A. 01-11031-PBS), available at <http://www.gnu.org/press/mysql-affidavit.pdf> (arguing that "Progress Software Corp. lost the right to distribute MySQL when it distributed NuSphere MySQL Advantage in a fashion that violated GPL"). Another enforcement action was filed around the time of publication of this Article. See Complaint at ¶ 13, *Andersen v. Monsoon Multimedia, Inc.*, No. 1:07-CV-

political message of software freedom carried by the license—enforcing the license is part of spreading the free software message.

Given the actual history of GPLv2 enforcement when the software is distributed in contravention of the license, this may appear to signal an orientation in the counterfactual to enforce the patented method embodied by licensing software under GPLv2. This may not necessarily be true for FOSS's primary rivals, however, because a third party might copy the software into a proprietary product and distribute it in violation of the copyright-based conditions of the GPLv2, yet not infringe the GPLv2 licensing method patent. The third party is not practicing the claimed method because the distribution is proprietary, i.e., it does not contain source code and/or require royalty payment(s). At its most potent, the patent will only confer potential control over licensing practices that are substantially similar to the claimed GPLv2 method.⁶⁷ Thus, the patent may be more useful against other strands within the FOSS movement than against pure proprietary software interests.

Fueled by its political cause, the world of information technology would not be surprised if the FSF enforced the GPLv2 patent, at least some of the time, and at least at the conversational level. It would do so to advance the agenda of free software. The FSF's conversational approach to GPLv2 license enforcement emphasizes correcting the offending software. Thus, a satisfactory remedy for the FSF has often been to discuss the problem with the entity that has incorporated GPLv2 licensed code into a proprietary product. This discussion aims to bring the code back into the open, making its source available and ensuring that it is not subject to proprietary software royalties.⁶⁸

Thus, asymmetries between the GPLv2 patent and copyleft limit FSF enforcement of the GPLv2 patent even if its actual copyleft enforcement suggests such enforcement. In copyright-based enforcement of GPLv2, a violation will often completely counter free software goals. In patent enforcement, the more the violator moves away from a GPLv2 licensing approach, the less likely patent infringement will be. It would go too far to say that GPLv2 provides enforcement power only over strands within the

08205-JES (S.D.N.Y. Sept. 19, 2007), available at <http://www.softwarefreedom.org/news/2007/sep/20/busybox/complaint.pdf> ("Defendant offers copies of the Firmware on its website, but does not offer any source code corresponding to the Firmware.").

67. Warner-Jenkinson Co. v. Hilton Davis Chem. Co., 520 U.S. 17, 39-40 (1997) (reciting the tripartite (triple-identity) test of finding equivalence under the doctrine of equivalents, where the accused infringing device has structure that meets a claim element/limitation non-literally by performing a substantially similar function in a substantially similar way with a substantially similar result); see also Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co., 535 U.S. 722, 732-33 (2002) (reaffirming the doctrine of equivalents as an alternative mode of asserting that a claim element is met for infringement purposes).

68. Vetter, *Exit and Voice in FOSS*, supra note 62, at 243-48.

FOSS movement, but the patent's potency is greater if it is closer to home. This is an important consideration informing much of the analysis below.

B. Abhorrence of Patents for Software versus Abhorrence of Patents for Licensing Methods

In evangelizing free software, the FSF advocates against patent protection for software.⁶⁹ Its advocacy is vocal and influential. The FOSS movement was an important factor in the European Union declining to pass a directive to clarify the scope of computer-implemented inventions. The worry was that the directive contained a Trojan horse—that by “clarifying” the law, the directive would have certified such patents as well as hastening their increase because the directive's language was too pliable.⁷⁰ The political economy of the entire affair is important and historical, but it is used here as an example to show the FSF's abhorrence of patent protection for software.⁷¹

This raises the question of whether this abhorrence transmutes into abhorrence for the somewhat different subject matter of a patent on a licensing method. If methods to achieve legal ends are patentable statutory subject matter, one can imagine injustices paralleling those the FSF sees from patenting software.

One way to test this reasoning is to reference an issue simmering at the time of this Article's publication: patent protection for tax planning methods.⁷² A primary philosophical argument against counting these as statutory subject matter is that citizens should not be forced to choose between infringing a patent and violating tax law (although it seems unlikely that all patent protection for tax planning methods would produce this result). The social injustice of this resonates with Richard Stallman's argument that a person needs control over her computer to live freely, and that

69. Free Software Found., Year End Appeal, http://www.fsf.org/index_appeal (last visited May 14, 2008) (noting that membership drive is to further various FSF causes, including its “defense against software patents”).

70. Nikki Tait, *European Position Is Left Patently Unclear*, FIN. TIMES, Sept. 21, 2005, at 11. The FOSS community was vocal and active in the European Union legislative process, including showing support for proposed amendments to favor FOSS licensing. *Id.* at 4; Committee on Legal Affairs and the Internal Market, *Report on the Proposal for a Directive of the European Parliament and of the Council on the Patentability of Computer-Implemented Inventions*, at 20, COM(2002) 92 final (June 18, 2003), available at <http://www2.europarl.eu.int/omk/sipade2?PUBREF=-//EP//NONSGML+REPORT+A52003-0238+0+DOC+PDF+V0//EN&L=EN&LEVEL=2&NAV=S&LSTDOC=Y> (“The rapporteur has also carefully weighed the arguments put forward by industry and the open source community . . .”).

71. Vetter, *Exit and Voice in FOSS*, *supra* note 62, at 248-55.

72. See generally Dan L. Burk & Brett H. McDonnell, *Patents, Tax Shelters, and the Firm*, 26 VA. TAX REV. 981 (2007).

the only true avenue to that control is through free software.⁷³ Whether Stallman's causal predicates are fully valid is not the critical point; rather, the fact that patent protection might give some individuals control over others' ability to do something as fundamental as obey tax laws might be considered an equivalent moral wrong.

The public/private distinction in law, however, takes the analysis in the opposite direction. It suggests that such philosophically based software patent abhorrence would not transmute against licensing methods. The licenses are private instruments among private parties, even though FOSS licenses are put forth in a generally applicable way. As such, patent protection in a FOSS licensing method, such as copyleft, is not likely to disable a citizen of her freedom to obey public law. This is not to say that the FSF would advocate such patent protection, but only to say that if it were available and common at the time of GPLv2's origination, its use would be consistent with the license's use of copyright. Perhaps at the time of GPLv2's origination Richard Stallman would have preferred to abolish all intellectual property protection in software. Since that alternative was not in his power, his second best solution, using copyright against its lineage, has registered impressive effectiveness.

The FOSS movement used a tool within its grasp, copyright protection in software, against another form of intellectual property, trade secret law, and to counter copyright itself.⁷⁴ If the tool of patent protection were also available, why not use it too, so long as it is not counterproductive to the values promoted by GPLv2? Perhaps the GPLv2 patent would not be repugnant to such values because the patent itself would be for the method embodied by practicing the license. As a right to exclude held by the FSF, it would be complementary to the copyleft conditions in GPLv2.

Changing the holder of the GPLv2 patent right, however, flips the argument. While it is likely benign to FOSS if held by the FSF, a GPLv2 patent held by Microsoft likely threatens the entire FOSS movement. This would be a reason for abhorrence of patents on licensing methods.

As is typical with counterfactual analysis, the question is more important than the answer. On the one hand, GPLv2 patent protection might be a

73. See Richard M. Stallman, *Why Software Should Not Have Owners*, in FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 47, 47-51 (Joshua Gay ed., 2002), available at <http://www.gnu.org/philosophy/fsfs/rms-essays.pdf> [hereinafter FREE SOFTWARE, FREE SOCIETY].

74. GPLv2 sought to eliminate secret source code during an era in the software industry when the dominant software revenue model was to keep source code secret, distribute object code, and often obtain contractual assent by the recipient to observe the secrecy. This, along with the baseline copyright protection that might attach to the software, founded the necessary exclusionary rights of "intellectual property" to charge royalties or license fees of some sort. GPLv2 needed to counter both secrecy and ongoing use royalties to frame a system where FOSS could be freely shared, changed, and shared again.

useful tool to further free software goals. That is the subject of the next Section. On the other hand, the exogenous nature of patent law's *in rem* right to exclude makes it a different creature from copyright. The patent right excludes anyone, even the independent developer, whereas copyright traces the lineage of copying from the right holder. If abhorrence of patents for software derives from institutional considerations that patent protection is a bad fit for software technology, a greater potential willingness to use patent protection for licensing methods would be the result, assuming such protection were available. However, if abhorrence of patents derives from concerns that its *in rem* system is too potent, this could underpin a philosophical rejection of the tool.⁷⁵

To conclude this Section, there is one additional technical point to make about the GPLv2 patent. Best practices would suggest that the GPLv2 license text be modified in light of the GPLv2 patent, granting permission to practice what the patent claims. Otherwise, the licensing method patent would be on a similar footing as software patents when distributors of GPLv2 licensed software hold patents covering the software. That footing is the unsettled law of implied license,⁷⁶ the details of which are put aside for this analysis. Revisions to the GPL in a later version eventually made the software patent license explicit.⁷⁷ In the counterfactual, however, the license text should grant permission to practice the GPLv2 patent. This concept will anchor the discussion of the first enforcement moment in Subsection III.D.1 below. That permission will state that a *prima facie* assertion of non-infringement of the GPLv2 patent arises when the unmodified GPLv2 license document is reapplied. In other words (using the words of patent law), licensing software using GPLv2 is a permitted embodiment of the process claimed by the GPLv2 patent.

Before visiting the enforcement moments in Section III.D, the next Section pauses to consider the other dominant license model within the FOSS movement: attribution-only licensing. This model deserves special treatment due to its prominence—it is non-infringing, but it does not threaten the validity of the GPLv2 patent. Moreover, its prominence constrains the enforcement power of the GPLv2 patent.

75. See Richard M. Stallman, *The Danger of Software Patents*, in FREE SOFTWARE, FREE SOCIETY, *supra* note 73, at 97, 97-98, 105 (describing the patent system disfavorably with an emphasis on its social costs).

76. See RAYMOND T. NIMMER, LICENSING OF INTELLECTUAL PROPERTY AND OTHER INFORMATION ASSETS 315 (2004) (“While courts often refer to the idea of an ‘implied license’ in litigation dealing with intellectual property claims, the content of the doctrine and its scope of application is far from settled and, often, seems far from coherent.”).

77. See GPLv3, *supra* note 10, § 11.

C. Non-Infringing Attribution-Only FOSS Licensing

The attribution-only license approach pre-dates GPLv2. Although many important FOSS projects operate under attribution-only licenses, these licenses merely claim copyright and then require that an attribution statement appear with the code. The attribution-only license does not have the features to ensure that the software remains transparent and shareable, although it often does so under institutional and practical influences. These licenses allow others to do practically anything with the software, including incorporating it into proprietary software, as long as there is notice that the software originated from the original project. These licenses do not even require that the source code be available, a key norm of the FOSS movement. Thus, attribution-only licenses are the least restrictive type of license used for FOSS projects.⁷⁸

The minimal licensing structure of an attribution-only license means that it does not fit within the claims of the GPLv2 patent. A rubric in patent law is that something that literally infringes if it comes later in time anticipates (and thus invalidates) a patent claim if it comes earlier in time. This demonstrates *apropos* that attribution-only licenses will not threaten the validity of the GPLv2 patent even though they are prior to it. The corresponding lack of GPLv2 patent enforcement power over attribution-only licensed software, however, is also of great consequence—it significantly limits the patent’s influence over much of the open source camp within the FOSS movement.

The next Section posits that the FSF has incorporated the tool of patent protection into its GPLv2 implementation and practices in the industry for most of the last two decades. Many of the items discussed below are particularly well known situations or events in the history of FOSS. Thus, they comprise a set of test cases to evaluate the effect of a FSF-owned GPLv2 patent. The treatment of each is not exhaustive: the goal is more to show the possible effects from enforcement than to answer the question of what happens in each hypothetical case.

D. Some Potential Enforcement Moments for the GPLv2 Patent

Enforcing an intellectual property right means considering the available remedies, which in United States patent and copyright law include

78. Given that attribution-only licenses do not require that the software be free of royalties, or that source code be available, there is some question as to whether attribution-only licenses are properly called FOSS. They are often categorized this way, however, because the programmers manage these projects using freely available source code and internet-based collaborative development.

damages and injunctive relief.⁷⁹ Patent law lacks statutory damages and thus has a different structure than copyright damages.⁸⁰ This difference could have great effect where the enforcer seeks damages, since actual damages or lost profits in a FOSS setting will raise interesting questions for software developed and shared freely.⁸¹ Certainly, some situations implicate actual damages, but others may have only a tenuous chance for this type of remedy. Thus, as between copyright and patent protection for the FOSS program (putting aside the counterfactual for the moment), copyright damages may provide greater leverage and threat value. The injunction question raises the need for a similar comparative inquiry, both for preliminary and permanent injunctions.⁸²

Given the potential modes of enforcement, the key consideration below is, under threat of or actual enforcement of the GPLv2 patent, whether the enforcement target diminished, altered, or ceased her activity. The analysis will not attempt to estimate how such diminution is derived from each component of some hypothetical remedial mix. This would be artificially over-precise, and is not essential to an analysis of the potential effects of the GPLv2 patent on FOSS's trajectory in the information technology ecosystem.

Finally, the enforcement discussion below generalizes infringement under U.S. patent law along a continuum, with literal infringement at one end and infringement under the Doctrine of Equivalents (DOE) at the other.⁸³ Though the situations and events described below are concrete and specific, differentiating between patent law's dual modes of infringement does not seem productive for every case. The primary point is that the DOE allows discussion of at least the potential for enforcement of the GPLv2

79. ROGER D. BLAIR & THOMAS F. COTTER, *INTELLECTUAL PROPERTY: ECONOMIC AND LEGAL DIMENSIONS OF RIGHTS AND REMEDIES* 12-13, 29-30, 69 (2005).

80. *Id.*

81. A complaint in a FOSS licensing enforcement action around the time of this Article requests damages, focusing on the interesting question of how to calculate them. *See* Complaint at ¶ 20, *Andersen v. Monsoon Multimedia, Inc.*, No. 1:07-CV-08205-JES (S.D.N.Y. Sept. 19, 2007), available at <http://www.softwarefreedom.org/news/2007/sep/-20/busybox/complaint.pdf>.

82. Beyond noting that the enforcement power of a patent or a copyright includes injunctive relief, the analysis will not attempt to further assess the comparative strength of either in a qualitative or quantitative way in the FOSS setting. *See eBay Inc. v. MercExchange, L.L.C.*, 547 U.S. 388, 390, 392, 393-94 (2006) (reaffirming that "the four-factor test historically employed by courts of equity" applies to issuance of a permanent injunction for patent infringement, overturning the action by the Court of Appeals for the Federal Circuit, which "articulated a 'general rule,' unique to patent disputes, 'that a permanent injunction will issue once infringement and validity have been adjudged,'" and noting that "[t]his approach is consistent with our treatment of injunctions under the Copyright Act").

83. *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 39-40 (1997); *see, e.g., Joshua D. Sarnoff, Abolishing the Doctrine of Equivalents and Claiming the Future After Festo*, 19 *BERKELEY TECH. L.J.* 1157 (2004).

patent's claims across a greater range of licensing method variation by third parties than if literal infringement were considered exclusively.⁸⁴

1. "Don't Change This License!" (it is copyrighted)

Most FOSS licenses say that third-party developers can apply the license to other software, but that the license may not be changed. Putting aside interactions with trademark law,⁸⁵ contract-based enforcement of the terms of the license, and any tort-based rights, these statements rely on copyright protection as it might attach to the license text as a literary work. For example, consider this quote from the Eclipse Public License:

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement.⁸⁶

The particular FOSS program referenced by this license, Eclipse, is managed by a foundation that also serves as the "Agreement Steward." IBM established this foundation as it moved into a supporting position for open source software.⁸⁷

The quoted language asserts that changing the license would be a copyright violation. However, the copyright protection in the Eclipse Public License may not be very potent due to a number of factors. Contract language has a functional character and thus often carries a minimal degree of expressiveness, as opposed to ideas—there is a need to use preexisting, standardized legal terms and phrases that constrain the alternatives in reciting the license rights.⁸⁸ Moreover, fair use is available under U.S. copyright

84. To speak of the DOE invites mention of the limitations on asserting that doctrine. These limitations must be ignored, however, because most of them require either a detailed understanding of the prior art, or the prosecution history, or both. The Article does not evaluate the former in a detailed way, and the latter cannot be generated in the abstract; it takes real patent prosecution to generate sufficiently detailed prosecution history.

85. The trademark issue is illustrated most explicitly by a FOSS license that is tied by name to a specific FOSS program or company. *See, e.g.*, Apple Computer, Inc., Apple Public Source License Version 2.0, ¶ 10 (2003), <http://www.opensource.apple.com/apsl> (discussing conditions for use of Apple's marks in association with the software). A developer revising the program and distributing something that is essentially no longer the original may create confusion among users of the technology if she associates her new software with any trademarks embedded in the license.

86. *See* Eclipse Public License—v 1.0, § 7, ¶ 4 <http://www.eclipse.org/org/documents/epl-v10.php> (last visited May 14, 2008).

87. Eclipse, About the Eclipse Foundation, <http://www.eclipse.org/org> (last visited May 14, 2008).

88. J.H. Reichman, *Charting the Collapse of the Patent-Copyright Dichotomy: Premises for a Restructured International Intellectual Property System*, 13 CARDOZO ARTS &

law, further limiting the control the license author may have over subsequent use of the license language.⁸⁹ For these reasons, there is reason to doubt the potency of copyright control over reuse of the license text.

The GPLv2 patent, however, would add another enforcement tool. In Section III.B above, it was noted that permission to practice the GPLv2 patent would use a *prima facie* approach—reapplying the unmodified GPLv2 license to software is non-infringement of the GPLv2 patent. Thus, the license text modifications that may not concern the modifier under copyright enforcement may require more heed due to the GPLv2 patent. This is because the patent's claim(s) will likely cover a wide range of revision to the license language as long as the license still implements the key techniques of GPLv2, such as source code availability, no royalties, and copyleft.

The more the revised license language deviates from the original, the more it may only “copy” ideas under copyright protection, and the greater the possibility it will not infringe the GPLv2 patent if the textual changes also represent a different mode of licensing. In this sense, the likelihood of both patent and copyright infringement decreases as the license reviser changes the text. For GPLv2, as compared to the Eclipse Public License, the analysis should treat the license preamble differently. The GPLv2 preamble is a core literary work under copyright. It is unique among FOSS license preambles, and is literary in a way few software licenses have ever been. Its copyright enforcement would be inherently more potent, and it would not implicate the GPLv2 patent because it does not embody the claim(s). The non-preamble sections of GPLv2 contain the provisions that embody the hypothetical claim.

The GPLv2 patent, then, would put teeth into the license's “don't change me!” provision.⁹⁰ The FSF would be able to more closely govern any close copies or minor changes to GPLv2. It would also be able to extend the *prima facie* approach to other FOSS licenses. This might enable the FSF to exert greater influence over a phenomenon often discussed as a problem in the FOSS movement—license proliferation.

ENT. L.J. 475, 495-96 (1995) (noting that courts provide thin protection for factual and functional works). *But see* 1-2 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2.18[E] (2007) (arguing that there is no basis for certain types of legal texts to not receive copyright protection and declining to mention that such protection would be characterized as “thin”).

89. 17 U.S.C. § 107 (2002).

90. GPLv2, *supra* note 2 (“Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.”).

2. License Proliferation

The license proliferation problem posits that too many FOSS licenses have been authored and made available. Disadvantages resulting from this situation would include difficulty for programmers in selecting from the increasing number of licenses, redundant licenses, “vanity” licenses,⁹¹ and dilution of the energy of a group associated with the open source camp that operates a certification mark for FOSS licenses.⁹² On the other hand, this congestion would provide an advantage to well-known licenses such as the GPLv2—programmers would rather choose it than search through a crowded field.⁹³

The FSF has extensive commentary and analysis of various FOSS licenses posted on its web site.⁹⁴ In its commentary, the FSF might favor some licenses over others because their provisions implement free software principles more pleasing to the FSF. This favoritism, assuming these are GPLv2-style⁹⁵ licenses, might specify that reapplying the favored licenses verbatim creates a *prima facie* non-infringement permission for the GPLv2 patent. Thus, the FSF’s program of license commentary could be buttressed by better legal status for some licenses.

The FSF license commentary undoubtedly influences some strands of the FOSS movement to prefer certain licenses. This is a first step against license proliferation; a published but unpopular license is less of a proliferation problem than one with a following. With the GPLv2 patent, the FSF might go further, and assert that its disfavored GPLv2-style licenses lack permission to practice the GPLv2 patent. Patent law allows the right-holder to exercise this discretion generally without constraint.⁹⁶ Thus, the FSF

91. OPEN SOURCE INITIATIVE, REPORT OF LICENSE PROLIFERATION COMMITTEE AND DRAFT FAQ (2006), <http://opensource.org/proliferation-report> (In the OSI’s efforts to reduce the number of FOSS licenses in circulation, its license proliferation committee characterized some licenses as “. . . specific to their authors and cannot be reused by others. Many, but not all, of these licenses fall into the category of vanity licenses.”).

92. Open Source Initiative, <http://www.opensource.org/index.php> (last visited May 14, 2008) (“Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.”).

93. Programmers also express their values by choosing the GPLv2 because it is the license most overtly associated with the FSF and its software freedom principles. See Vetter, *Exit and Voice in FOSS*, *supra* note 62, at 202-04, 221-24.

94. Free Software Found., Licenses, <http://www.fsf.org/licensing/licenses> (last visited May 14, 2008).

95. A GPLv2-style license is one that imitates the GPLv2 in its key characteristics of source code availability, no royalties, and copyleft.

96. Most of the law potentially constraining a patent holder’s right to wield the instrument, such as antitrust, is inapplicable to the counterfactual or does not add to the analysis, nor do more mundane matters such as whether particular acts of infringement fall

could limit its approval to licenses that are worthy of the principles of free software.

This might create interesting interactions with the Open Source Initiative (OSI), the group responsible for operating the certification mark referred to above, called the Open Source Definition (OSD). The OSD defines criteria against which the OSI evaluates and certifies licenses.⁹⁷ The OSD license criteria share many similarities with GPLv2. The OSI categorizes GPLv2 as an “open source” license. One difference, however, is that unlike GPLv2, the OSD does not require that a license demand that modifications be distributed under the same terms. The OSD merely says that a license must allow such a condition, but that a license need not have it.⁹⁸ The OSD itself is not a FOSS license, but only a measuring device used to classify FOSS licenses.

Thus, the OSD criteria would certify many licenses that are too different from the GPLv2 patent to fear infringement, particularly many licenses that take an attribution-only approach. A FOSS license that does not require copyleft or reapplication of the same terms, or even reapplication of the critical FOSS-preserving terms, would not literally infringe the GPLv2 patent. It also likely would not infringe under the DOE, since the licensing function and the licensing result are likely not substantially similar, or, in the other DOE formulation, insubstantially different.⁹⁹ Thus, the GPLv2 enforcement power would not reach many OSD certified licenses.

Assume, however, that the OSI wanted to certify and list a license that embodied the GPLv2 patent claim(s), but also embraced Digital Rights Management (DRM) to enforce the FOSS license. Given the FSF’s stance against DRM as expressed in version three of the GPL,¹⁰⁰ it is conceivable that the FSF would withhold permission for the DRM-embracing license to practice the GPLv2 patent. Perhaps as a result, the OSI would choose not to list the license, though this is unlikely since a certification mark system un-

within patent law’s statute of limitations for bringing an infringement action. 35 U.S.C. § 286 (2000).

97. Open Source Initiative, <http://www.opensource.org> (last visited May 14, 2008) (“The Open Source Initiative (OSI) is a non-profit corporation formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open-source community.”).

98. Open Source Initiative, The Open Source Definition § 3, <http://www.opensource.org/docs/osd> (last visited May 14, 2008).

99. Warner-Jenkinson Co. v. Hilton Davis Chem. Co., 520 U.S. 17, 38-41 (1997) (noting that two linguistic versions of the test for the doctrine of equivalents are in use and declining to elevate either over the other, where one uses the construct of insubstantial differences, while the other, the triple-identity test, uses the construct of substantial similarity for function, way, and result).

100. GPLv3, *supra* note 10, § 3.

der trademark law obligates nondiscriminatory treatment.¹⁰¹ Thus, proliferation is diminished but not eliminated. The DRM-embracing license may be listed, but the FSF is likely to let the world know that no one has permission to practice it.

In sum, the GPLv2 patent would likely be a minor influence on license proliferation because it would only cover GPLv2-style FOSS licenses. This influence would tend to reduce the number of FOSS licenses. In addition, any GPLv2-style licenses would likely pay greater heed to the free software agenda. Even if these influences are slight, they could have impacts in the ongoing strategic game occurring both within the FOSS movement and external to it in information technology.¹⁰²

The next two enforcement moments look at slight variations of the GPLv2 that will be assumed to fall within the patent's claims either literally or as an equivalent, thus establishing the FSF's patent enforcement power.

3. *GPLv2 as Applied to the Linux Kernel*

The Linux kernel is central to the GNU/Linux operating system distributions. Each distribution is not a single program, but an inter-functioning collection of many components from multiple sources, with different FOSS licenses. Thus, virtually every component will interact with the kernel. Moreover, a user might install a GNU/Linux distribution and run proprietary software on the operating system. Such proprietary software also interacts with the Linux kernel. This raises an issue: if the kernel is licensed under GPLv2, do such interactions mean that GPLv2 must apply to the proprietary software?

This issue arises because the GPLv2 has some provisions that are commonly called the license's "viral" or "infectious" characteristics. What is at stake is whether the GPLv2 terms must be extended to other software when one couples or intermixes that other software with the GPLv2-licensed software and distributes the resulting "whole." The word "whole"

101. 15 U.S.C. § 1064(5) (2000) (Lanham Act § 14(5), providing that a certification mark is at risk of cancellation if the markholder "discriminately refuses to certify or to continue to certify the goods or services of any person who maintains the standards or conditions which such mark certifies"); *see also* 3 J. THOMAS MCCARTHY, MCCARTHY ON TRADEMARKS AND UNFAIR COMPETITION § 19:90 (4th ed. 2008). The OSI could change its standards to only list licenses not under infringement risk from the GPLv2 patent, but this would directly cede control over a portion of the certification process to the FSF. Framing the addition to the certification standard in this way places full discretion with the FSF. It can use its own criteria to determine whether to pre-commit to permit a particular license to practice the patent. Finally, listing licenses with GPLv2 patent infringement risk raises in some remote way the worry about indirect patent infringement liability for the OSI. In United States patent law, there are two types of indirect liability: inducement and contributory liability. 35 U.S.C. § 271(b)-(c). Noting this, however, is as far as the analysis will go on the point.

102. *See* WEBER, *supra* note 7, at 134-49.

is emphasized because it is one of the operative words in GPLv2.¹⁰³ It is ambiguous under GPLv2 whether the reach of this characteristic ends with what copyright law considers a derivative work, or extends further, although in that case, contract enforcement might be necessary. Other articles have treated these issues,¹⁰⁴ so the minimal treatment here will simply explain why proprietary software vendors might worry about running on GNU/Linux. Proprietary software vendors control their source code and usually keep it secret, so they might have to revise their software in order for it to run on GNU/Linux. After doing so, however, they would not want a free software advocate claiming that they must now release their proprietary software as FOSS under GPLv2.

There are a variety of technical and copyright-related reasons why such a claim would be tenuous. Nevertheless, the claim would be complicated by the availability of the source code for the Linux kernel. Operating system kernels, including Linux, provide an interface, often called the system calls, which the other programs running on the operating system use to interact with the kernel.¹⁰⁵ Linux kernel source code availability allows a proprietary software vendor to go below the system call interface and use its lower-level constructs directly. In a proprietary operating system such as Microsoft's Windows, on the other hand, similar constructs exist within its kernel, but are hidden and thus effectively unusable. Using such constructs in Linux is a more intimate degree of intermixing, and heightens the risk that the GPLv2's infectious provisions reach the "whole."

GNU/Linux has been tremendously successful in the server computing market, where much of the important non-Microsoft proprietary software is available for it. The success of any operating system is dependent in part on the number of applications available for it, so anything that diminishes the universe of available applications inhibits growth.

Even though the "infectious" claim is tenuous as between the kernel and the proprietary application, a slight modification to the GPLv2 as applied to the Linux kernel helps to ease the worry. Upon adopting GPLv2 for the Linux kernel, Linus Torvalds modified the license's permission set with a clarifying statement carried by a file in the software.¹⁰⁶ The statement says

103. See GPLv2, *supra* note 2, § 2.

104. See generally McGowan, *supra* note 7, at 289-302 (examining the FOSS approach and contract law implications); Vetter, *Infectious OSS*, *supra* note 8, at 129-30 (discussing the feature of the GPL oftentimes referred to as its "viral" characteristic).

105. Some readers will be familiar with the term application programming interface (API), which is a collection of functions or procedures that one process (a program in execution on an operating system) can use to ask another process to do some computing task for the requestor. The term is associated with older software technology for program to program interaction, but one can think of the kernel system calls as an API for the kernel.

106. Linus Torvalds, *The Linux Edge*, in OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 7, at 101, 109.

that any program that uses the kernel through its “normal system calls” is, in effect, deemed not to be part of the “whole,” even if copyright’s derivative work right would reach that far. This pragmatic action is characteristic of Linus Torvalds, and pragmatism undertones the approach of the open source camp. While it is impossible to empirically evaluate the importance of this, the “normal system calls” proviso almost certainly helped the adoption of GNU/Linux in the marketplace. The proviso gave proprietary software vendors greater confidence in porting their software to GNU/Linux.

In the counterfactual, however, Torvalds’s revisions would have infringed the GPLv2 patent. The license variation is small, so licensing the kernel under the GPLv2 with the “normal system calls” proviso would likely constitute an infringing embodiment of the patent claim(s). The market success of the Linux kernel was unanticipated, so by the time it caught the attention of the FOSS movement, that very success was Linus Torvalds’s best defense. Any successful FOSS application would help the movement. Using the GPLv2 patent to force Torvalds to remove the “normal system call” proviso would only interject a worry that might slow the Linux kernel’s growth at a time when it was clearly becoming the center of the most widespread FOSS success.¹⁰⁷ This strategic reality would likely dominate any FSF desire to purify GPLv2 as applied to the Linux kernel.

On the other hand, besides the differing philosophical approaches to FOSS between the two camps, for a time there was an effort within the FSF to create its own operating system kernel.¹⁰⁸ The success of Linux effectively eliminated this need over time. Would this kernel development competition have raised the possibility of enforcement? The question is inestimable, but it is one example of other strategic forces that might have influenced the FSF’s enforcement preference.

4. “File-Level” Weak Copyleft

Like the Linux kernel, the next enforcement moment involves a famous situation within the FOSS movement, this time regarding the development of web browsing technology for the Internet. In the 1990s, a company called Netscape was the market leader in web browsers, but saw its

107. A common approach in the information technology industry is for competitors to quietly promote “Fear, Uncertainty, and Doubt,” or “FUD” among the customer base. The FUD is typically designed to disadvantage customer perceptions of the competitors of the FUD-spreader. In the context of GNU/Linux, the general FUD phenomenon has manifested itself with proprietary software vendors sometimes pointing to the strong copyleft feature of GPLv2 as a reason for worry. One example of such hypothetical worry is that porting a proprietary application to GNU/Linux puts it at risk for “infection” by the GPLv2 terms because it is now running on a kernel covered by that license, or interoperating with other components of GNU/Linux covered by that license.

108. See MOODY, *supra* note 3, at 26.

position under attack when Microsoft incorporated a web browser into its operating system. One of Netscape's eventual strategic responses was to convert its web browser into a FOSS project, originally named Mozilla, and later stewarded by a foundation with that same name.¹⁰⁹ The conversion was both a programming and a legal affair.¹¹⁰ The project engendered new FOSS licenses, and many imitations followed.

One such license, the Mozilla Public License (MPL),¹¹¹ implements what is sometimes called "weak" copyleft as compared to GPLv2 because MPL uses a technical bright line to express when intermixed software must be distributed under MPL's terms. The scope of GPLv2's reach is at least as far as copyright's derivative work right, but MPL stops short of that. MPL states that if a file is modified, then the new code put into that file must be under MPL.¹¹² However, if the new code is partitioned into a separate file, it need not be licensed under MPL.¹¹³

Whether the GPLv2 patent would provide enforcement power against MPL is an interesting question that contrasts literal infringement with infringement under the DOE. The MPL technical bright line to cabin the scope of copyleft is a notable difference and might not be an infringing equivalent. Under the DOE's tripartite test, the MPL copyleft boundary may not be substantially similar. GPLv2 uses copyright's derivative work right to draw a fuzzy boundary, while MPL uses a technical concept to draw a bright line. This suggests that patent law's DOE analysis would consider them not to be substantially similar functions, implemented in a substantially similar way with substantially similar results. Intermixing new code with MPL-licensed code while avoiding MPL coverage on the new code is much more plausible with MPL than with GPLv2.

Accepting this logic, the GPLv2 patent would minimally impact the Mozilla project. The project itself was not a great success in resurrecting Netscape's commercial prospects, but the foundation became an important fixture in the FOSS movement. It was central to the highly successful Fire-

109. See Mozilla, About Mozilla, <http://www.mozilla.org/about> (last visited May 14, 2008); Jim Hamerly & Tom Paquin, *Freeing the Source: The Story of Mozilla*, in OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 7, at 197, 203-06 (describing the events leading up to Netscape's decision to release the source code for its web browser, Mozilla).

110. Bruce Perens, *The Open Source Definition*, in OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 7, at 171, 184 (describing Netscape's release of source code for its Web browser and the legal analysis of the source code necessary for release).

111. Mozilla Public License Version 1.1, <http://www.mozilla.org/MPL/MPL-1.1.html> (last visited May 14, 2008).

112. *Id.* § 1.9.

113. *Id.*

fox open source browser.¹¹⁴ Later in the movement, during the 2000s, a new approach, the dual license, gained prominence for commercial deployment of FOSS programs.

5. *Dual Licensing and Patent Law's "Extra Element" Infringement Rule*

A dual license seeks to establish two paths for a program, a FOSS path and a commercial path. Each program path can cross-fertilize the other with code, and the commercial path commits to certain benefits for the FOSS path. Companies typically support a dual license, seeking to foster a community of developers, to contribute to that community, and to harvest from the community some benefit to deploy in the program's commercial path. This structure means a meta-license surrounds the FOSS license. When successful, the result is that a for-profit company becomes a nexus for a community of developers, many of whom are not otherwise affiliated with the company, but who trust the company to further FOSS movement goals, along with commercial objectives.

If the FOSS license path of the dual license were considered a GPLv2-style license embodying the GPLv2 patent claim(s), this would raise the question of infringement using what is sometimes called the "extra element" rule in patent law for infringing articles or processes.¹¹⁵ For a product claim, an article infringes if it has the cooperative combination the claim recites. It typically does not matter if the article has an extra component, since that "extra element" does not remove the fact that the article has what the claim recites.¹¹⁶ Similarly, for a process claim, an extra step typically will not ward off infringement if the other allegedly infringing steps embody the claimed process.

If the meta-license that puts in place a parallel commercial path for the FOSS software is considered an extra step or element, then the GPLv2 patent enforcement power might be able to control or prohibit any dual licensing using a GPLv2-style FOSS path. This analysis would be sensitive to both the claim language of the GPLv2 patent and the license text of the dual license. Thus, some situations might not produce infringement. However, there is significant risk that many dual licenses will infringe.

Most companies developing FOSS around a dual license approach would fall into the open source camp, even if using GPLv2 for the FOSS

114. Mozilla, Firefox 2, <http://www.mozilla.com/en-US/firefox> (last visited May 14, 2008).

115. See MUELLER, *supra* note 53, at 64-65.

116. The "extra element" rule applies for infringement when the claim is open-ended, meaning that its transition word is "comprising" or is one of the standard synonyms used to signal such a claim. See *id.* at 64-65.

path. The developer community often prefers that the company use GPLv2 because the license has a strong reputation among programmers.¹¹⁷ The programmers appreciate the fact that the FOSS side of the dual license will adhere to some degree of free software principles, even if the company is not a non-profit organization devoted to free software. There may be some tendency within the free software camp to see dual licensing as too great a compromise of free software principles, due to the fact that dual licensing may result in software being distributed under the traditional proprietary model. Sometimes the proprietary version is a complete duplicate of the version available under the FOSS license, but the end-users desire training, support, warranties, and other services only available from a commercial company. Ensuring that all of the commercial-path code is available on the FOSS path, however, may lessen the offense to free software principles. Nevertheless, sometimes the commercial distribution blends in other proprietary software. Because of this, the FOSS path of the dual license might be seen to facilitate non-free programs. This would resonate with other situations where the FSF might use its GPLv2 enforcement power.

The enforcement threat to dual licensing is important because the other dominant type of FOSS licensing has less need for dual licensing. Attribution-only licenses are sufficiently permissive that while pragmatic considerations might move a commercial company to support the development community, license terms inhibiting the software's commercial use and distribution are minimal. Thus, a commercial company may find it difficult to develop the community under an attribution-only license. The community wants the pre-commitment of a dual license where one path is GPLv2, or a similar license that guarantees a free software path. In other words, the place where the need for dual licensing is greatest is also where the GPLv2 patent enforcement power would be most potent.

The importance of dual licensing to the FOSS movement is an important related question. At the time of this writing, however, it is not as important as, for example, the success of GNU/Linux. If dual licensing's importance were to grow, the GPLv2 patent could potentially alter the trajectory of the movement, at least in the counterfactual. In considering the last enforcement moment, one part of that trajectory includes surprising developments by Microsoft to involve itself with open source licenses and software development approaches.

117. See David McGowan, *SCO What? Rhetoric, Law and the Future of F/OSS Production* 33-34 (Univ. of Minn. Law Sch., Research Paper No. 04-9, 2004), available at <http://www.ssrn.com/abstract=555851> (suggesting that the GPL terms may not necessarily be optimal for the developers who use them, but are employed in a trademark sense as a quasi brand identity espousing certain development procedures or ideological beliefs developers may find more important than the terms themselves).

6. Microsoft's Shared Source Licenses

After Microsoft took appreciable note of the FOSS movement, its posture toward it was, and remains in most ways, adversarial. This is particularly true for the free software strand of the movement.

In the second decade of FOSS, however, a variety of forces led to a slight tilt in that posture. The FOSS movement grew in stature and success, particularly for server computing and the Internet. Microsoft's maturation led to market dominance in key areas of desktop computing, but the company also came under scrutiny from antitrust and competition authorities in the Americas and in Europe. In addition, Internet-enabled changes in the information technology ecosystem, heightened pressures for interoperability, technological transparency and standards, and gave desktop prominence to network computing applications such as Google's Internet-wide search.

Against the backdrop of these forces, in the mid-2000s, Microsoft began a "shared source" program that initially allowed customers to view source code. This was evidence that, because of the FOSS movement, the market was beginning to recognize the transparency and interoperability value in open code. Currently, this program operates under a "Reference License" promulgated by Microsoft.¹¹⁸ Shared source at Microsoft, however, has grown to include two other licenses, one of which relates to the attribution-only category of Section III.C above, and thus would certainly not infringe the GPLv2 patent.¹¹⁹ The other license is the Microsoft Reciprocal License.¹²⁰ Using generic versions of these licenses made available by Microsoft, the company claims that third-party developers have distributed

118. Microsoft, Microsoft Reference License (Ms-RL) (Mar. 8, 2007), <http://www.microsoft.com/resources/sharedsource/licensingbasics/referencelicense.aspx> (last visited May 14, 2008). Microsoft's description of the Reference License resonates with transparency and interoperability values:

The license prohibits all use of source code other than the viewing of the code for reference purposes. The intent of this license is to enable licensors to release, for review purposes only, more sensitive intellectual property assets.

Microsoft commonly uses this license for developer libraries where modification is not required to make use of the source code. In these cases, the importance of transparency is based on the need for developers to more deeply understand the inner workings of the source code. In doing so, the licensees will be more effective in writing software that makes use of the licensed source code.

Microsoft, Shared Source Licenses (Oct. 18, 2005), <http://www.microsoft.com/resources/sharedsource/licensingbasics/sharedsourcelicenses.aspx> [hereinafter SharedSource] (last visited May 14, 2008).

119. Microsoft, Microsoft Public License (Ms-PL) (Oct. 12, 2006), <http://www.microsoft.com/resources/sharedsource/licensingbasics/publiclicense.aspx> (last visited May 14, 2008).

120. Microsoft, Microsoft Reciprocal License (Ms-RL) (Oct. 12, 2006), <http://www.microsoft.com/resources/sharedsource/licensingbasics/reciprocallicense.aspx> [hereinafter MS Reciprocal License] (last visited May 14, 2008).

over hundreds of source code distributions.¹²¹ Thus, at some level, Microsoft is now facilitating open source software.¹²²

Under the Microsoft Reciprocal License, this facilitation includes reciprocity for source code disclosure. The license does not prohibit royalties. The reach of the Microsoft Reciprocal License's reciprocal or copyleft provision is similar to the MPL "file-level" copyleft mechanism discussed in Subsection III.D.4 above—under the Microsoft Reciprocal License, changes to a file are the determining factor in considering whether the license must extend to the intermixed code.¹²³

Given that the analysis of "file-level" copyleft posits at best a tenuous infringement scenario under the DOE, the Microsoft Reciprocal License is a step further away from infringing the GPLv2 patent because it does not prohibit royalties. Users may commercialize the code, so long as they make source code available.

Even if the infringement case is more tenuous, the target is more tempting. The free software strand of the FOSS movement, and the FSF in particular, are likely suspicious of Microsoft and dubious about any real commitment to open source principles, much less free software ideals. Thus, a reasonable strategic response, if there were any GPLv2 enforcement power, would be to disfavor the Microsoft Reciprocal License, or to outright declare that no permission was granted to practice the GPLv2 patent by licensing under the Microsoft Reciprocal License. One strategic justification for this from the free software perspective is that keeping Microsoft at a distance prevents infiltration. Under this notion, the FOSS movement is disserved if developers flock in droves to the Microsoft open source portal¹²⁴ and expend energies there as opposed to other general community-

121. See SharedSource, *supra* note 118 ("These non-Microsoft developers have used the Microsoft Shared Source licenses to license more than 700 of their own source-code distributions."). Whether that number is significant or not is a matter of opinion, but one way to calibrate it is by comparison to the popular Sourceforge FOSS repository, which around the time of this Article, listed nearly 200,000 projects. See Sourceforge.net, <http://sourceforge.net> (last visited May 14, 2008) ("Registered Projects: 173,134 Registered Users: 1,818,459.").

122. Microsoft, Open Source at Microsoft: World of Choice, <http://www.microsoft.com/opensource/choice.mspx> (last visited May 14, 2008) ("Microsoft is continually growing the number of products released with open source access.").

123. See MS Reciprocal License, *supra* note 120, § 3(A). Microsoft's commentary about the Reference License notes:

Thus, the intent of the reciprocal license is to use licensing as a mechanism to keep certain community-based code "in the community," while allowing companies to commercialize and license (under terms of their choice) their "value add" code that interacts with the community-based code.

SharedSource, *supra* note 118.

124. Microsoft, Open Source at Microsoft, <http://www.microsoft.com/opensource/default.mspx> (last visited May 14, 2008).

sponsored projects. Microsoft's shared source efforts would be painted as a shrill ploy to channel some of the FOSS movement away from genuine free software development. Whether this is in fact Microsoft's motivation, or whether more benign influences such as customer demand for interoperability and transparency motivate it, is irrelevant. In the counterfactual, Microsoft's true motivation ultimately does not matter because the GPLv2 enforcement power is unlikely to reach Microsoft's "file-level," royalty-neutral, copyleft license.

The Microsoft Shared Source enforcement moment, then, would be yet another example of the GPLv2 patent's inability to reach the proprietary software world. The most potent FOSS enforcement would continue to be the unique copyleft conditions rooted in copyright law and enabled by the high desirability and quality of many FOSS programs.

CONCLUSION

This last enforcement moment ends the counterfactual where the FOSS movement began—in opposition to proprietary software. The copyright-based licensing system that has lifted FOSS through the last two decades remains a much more potent tool against the enemy of the free software strand than the hypothetical GPLv2 patent would be. Patent law's right to exclude would give notable, but not overwhelming, enforcement power within the movement, and little if any power outside of it.

Although the counterfactual is hypothetical, contemporaneous with this Article's publication, there are patent applications pending before the PTO claiming software licensing methods directly related to FOSS licensing.¹²⁵ These applications were published under U.S. patent law's eighteen-

125. Authenticating Licenses for Legally-Protectable Content Based on License Profiles and Content Identifiers, U.S. Patent Pub. 20050125358 (filed Dec. 4, 2003) (disclosing a license-authority method of analyzing license information related to intellectual property, such as software, by evaluating license information from both a content owner and a content user to ensure that the content user's resulting license information is authentic and accurately represents the restrictions imposed by the content owner's license restrictions); Software License Isolation Layer, U.S. Patent Pub. 20040068734 (filed Oct. 7, 2002) (disclosing a system and method for using a license isolation layer in the process of software development to detect conflicting license terms); Integrated Development Environment for Managing Software Licensing Restrictions, U.S. Patent Pub. 20060242077 (filed Apr. 21, 2005) (disclosing a system and method for identifying software license conflicts during software development to prevent licensing conflicts by identifying a target software license associated with a software development project, identifying a license associated with the code object, and allowing the code object to be included in the software development project only if the license associated with the code object is compatible with the target software license); Methods and Apparatuses for Reviewing General Public Licenses, U.S. Patent Pub. 20060288421 (filed June 15, 2005) (disclosing a method and apparatuses for detecting conflicts in licensing terms between a first general public license corresponding to a first group of code and a second general public license corresponding to a second group of code);

month publication requirement,¹²⁶ and one has even issued.¹²⁷ Many of these applications focus on the “infectious” provision of licenses like the GPLv2. Generally, they claim methods and structures to segment and partition software so that a strong copyleft mechanism such as that found in the GPLv2 will not reach intermixed software. These applications for copyleft-immunization patents illustrate the role patent protection may play in the future of FOSS licensing methods, above and beyond the already notable impact software patents have had on information technology.

As a postscript, the counterfactual needs one final annotation. Assuming that the term is measured starting in 1991, in a few short years from the time of this writing, in 2011, the GPLv2 patent will expire. Twenty years is a long time compared to the life span of many technologies in computing, but perhaps twenty years will be a short time in the lifespan of the FOSS movement.

Automated License Dependency Resolution and License Generation, U.S. Patent Pub. 20020188608 (filed June 12, 2001) (disclosing a system for identifying potential license issues when combining software components and identifying potential licenses to resolve these issues); Resolving License Dependencies for Aggregations of Legally-Protectable Content, U.S. Patent Pub. 20050125359 (filed Dec. 4, 2003) (disclosing a method of identifying and comparing license attributes from two protectable contents to determine license attributes for a new protectable content based on a combination of the two protectable contents).

126. 35 U.S.C. § 122(b) (2000).

127. Method and System for Managing Intellectual Property Aspects of Software Code, U.S. Patent No. 7,277,904 (filed Dec. 18, 2003) (issued Oct. 2, 2007) (disclosing a system for managing license terms for a large-scale software project).