COMMERCIAL FREE AND OPEN SOURCE SOFTWARE: KNOWLEDGE PRODUCTION, HYBRID APPROPRIABILITY, AND PATENTS

*Greg R. Vetter**

INTRODUCTION

Robert Jacobsen might seem like a party with the equities and the law in his favor. After all, he provided his software to the world under a royalty-free license allowing broad public use.[1] The license even allowed companies to use the software in proprietary products with minimal nonpecuniary conditions.[2] He led a group of hobbyists who used the software to enjoy creating model railroads.[3] A company[4] competed with these volunteers who freely shared their work. The company used some of Jacobsen's software in its commercial, royalty-bearing product, but it did not follow the articulated license conditions. Under dominant notions of copyright licensing, the company faced a risk of infringement of the

* Associate Professor of Law, University of Houston Law Center; Co-Director, Institute for Intellectual Property and Information Law; 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 Law Center students Justin Bronn, Domingo Llagostera, and Bo Tang for excellent research assistance. For helpful comments and discussion, I thank Paul Janicke, Lee Ann Lockridge, Shubha Ghosh, Mike Madison, Brett Frischmann, Sarah Rajec, participants at the George Washington University Law School Spring 2009 Intellectual Property Workshop Series by the Dean Dinwoodie Center for Intellectual Property Studies, participants at the Drake University Law School 2009 Intellectual Property Scholars Roundtable by the Drake Intellectual Property Law Center, and participants at the *Fordham Law Review*'s Fall 2008 Symposium: *When Worlds Collide: Intellectual Property Laws at the Interface Between Systems of Knowledge Creation*, held October 31 and November 1, 2008, at the Fordham University School of Law.

1. Jacobsen v. Katzer (*Jacobsen II*), 535 F.3d 1373, 1376–77 (Fed. Cir. 2008).

2. Jacobsen v. Katzer (*Jacobsen I*), No. C 06-01905, 2007 WL 2358628, at *6 (N.D. Cal. Aug. 17, 2007); *see also* Open Source Initiative, Artistic License 2.0, §§ 3–5, *available at* http://www.opensource.org/licenses/artistic-license-2.0.php. Robert Jacobsen's group originally issued its software under the Artistic License. The Artistic License is one of many dozens (or, perhaps, hundreds) of licenses that would fall into the broad category of licenses for free and open source software (FOSS).

3. *See Jacobsen I*, 2007 WL 2358628, at *1 ("Plaintiff, Robert Jacobsen . . . is a professor of physics . . . as well as a model train hobbyist and a leading member of the [Java Model Railroad Interface (JMRI)] Project."); *see also* JMRI: A Java Model Railroad Interface, http://www.decoderpro.com (last visited Mar. 23, 2009).

4. *See* KAM Industries, About KAM, *originally available at* http://www.trainpriority.com/Aboutus/AboutKam.aspx (last visited Oct. 10, 2008) (on file with author).

reproduction right, especially since any fair use arguments in relation to Jacobsen and his group would by no means eliminate the company's infringement risk.

The U.S. District Court for the Northern District of California surprisingly did not find in favor of Jacobsen. The result was reversed in Jacobsen's favor at the appellate level at the U.S. Court of Appeals for the Federal Circuit. After receiving a letter from the company threatening him with patent infringement, Jacobsen filed suit seeking declaratory judgment that the patent was invalid, unenforceable, and noninfringed. In the suit he also brought copyright claims. The appeal involved only the copyright issue of whether the company's noncompliance with the public license conditions raised a copyright claim. The appellate court held that it did, and that copyright remedies, including the possibility of a preliminary injunction, applied.[5] In its analysis, the appellate court spoke approvingly of the beneficial nature of software development under public licenses.[6]

Compare Robert Jacobsen to MetaCarta, a company involved in both proprietary software development and related services for its users, and involved with certain niche open source communities relating to software for displaying geographic information.[7] I choose MetaCarta as a stylized example because it is not involved in any litigation of which I am aware.[8] But it has a noteworthy approach to its role in the greater world of free and open source software (FOSS) development. MetaCarta contributes some of

5. On remand from the U.S. Court of Appeals for the Federal Circuit, the U.S. District Court for the Northern District of California denied Jacobsen the preliminary injunction. Jacobsen v. Katzer (*Jacobsen III*), No. C 06-01905, 2009 WL 29881, at *7–10 (N.D. Cal. Jan. 5, 2009) (denying preliminary injunction to Jacobsen on his copyright-based claims under the reasoning that his harms were too speculative on the record before the court).

6. *Jacobsen II*, 535 F.3d at 1378–79 ("Open source licensing has become a widely used method of creative collaboration that serves to advance the arts and sciences in a manner and at a pace that few could have imagined just a few decades ago."); *see also* Wallace v. IBM Corp., 467 F.3d 1104, 1105 (7th Cir. 2006) ("Copyright law, usually the basis of limiting reproduction in order to collect a fee, ensures that open-source software remains free: any attempt to sell a derivative work will violate the copyright laws, even if the improver has not accepted the GPL. The Free Software Foundation calls the result 'copyleft.'"). To be entirely clear about the essence of the case, whether the General Public License (GPL) was price fixing, in *Wallace v. IBM*, Judge Frank Easterbrook concludes as follows: "The GPL and open-source software have nothing to fear from the antitrust laws." *Id.* at 1108.

7. *See* MetaCarta, About Us: Company Overview, http://www.metacarta.com/about-us-overview.htm (last visited Mar. 23, 2009) (noting that its "products make data and unstructured content 'location-aware' and geographically relevant").

The open source communities supported by MetaCarta each have their own web site. *See, e.g.*, FeatureServer—RESTful Geographic Feature Storage, http://featureserver.org (last visited Mar. 23, 2009); OpenLayers: Free Maps for the Web, http://openlayers.org (last visited Mar. 23, 2009) ("MetaCarta developed the initial version of OpenLayers and gave it to the public to further the use of geographic information of all kinds. OpenLayers is completely free, Open Source JavaScript, released under a BSD-style License."); TileCache—Web Map Tile Caching, http://tilecache.org (last visited Mar. 23, 2009) ("TileCache is . . . made available under the BSD license by MetaCarta.").

8. I did not know of MetaCarta before this essay. I came across the company in my research seeking examples of unusual juxtapositions of open source development and use of patents.

its software to the FOSS community by acting as the organizing hub for three FOSS projects. This is not unheard-of. More uniquely, however, it also actively seeks a small portfolio of patents in related areas of software technology. Following a trend, MetaCarta is backed by venture capital investors while explicitly embracing the FOSS movement. Compared to a for-profit entity such as MetaCarta, Robert Jacobsen is a sympathetic figure for a court. He was a volunteer developing FOSS with public benefit spillovers.[9] His motivations likely fit within some of the typically offered explanations for FOSS volunteerism: to scratch a technological itch; to have fun; to participate in a community; to learn; or to enhance career prospects. MetaCarta's motivations are those of a for-profit firm with investors hoping for return and market share. While software patenting has become common among information technology companies, much of the FOSS movement would see it as nonbeneficial. MetaCarta, however, represents a trend: "commercial FOSS"[10] that hybridizes proprietary software appropriation techniques with conventional FOSS volunteerism-centric development.

Jacobsen and MetaCarta illustrate a dualism in FOSS that channels the knowledge production and distribution influences of the movement and could impact the perspective of future courts as they engage other licensing law issues likely to arise. Court interpretation of FOSS licenses on their merits and against the backdrop of intellectual property and information licensing law can enhance or diminish the efficacy of the FOSS licensing movement.[11] If the district court's analysis in *Jacobsen v. Katzer*[12] had

---

9. Jürgen Bitzer & Philipp J. H. Schröder, *The Impact of Entry and Competition by Open Source Software on Innovation Activity*, *in* THE ECONOMICS OF OPEN SOURCE SOFTWARE DEVELOPMENT 219, 228 (Jürgen Bitzer & Philipp J. H. Schröder eds., 2006) (differentiating the greater degree of knowledge spillover resulting from open source software in comparison to proprietary licensed software); Brett M. Frischmann & Mark A. Lemley, *Spillovers*, 107 COLUM. L. REV. 257, 274–79 (2007) (discussing innovation spillovers and reasons why such may not need full internalization to the innovator yet remain beneficial for her); *see also* Eric von Hippel & Georg von Krogh, *Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science*, 14 ORG. SCI. 209, 213–15 (2003) (arguing that FOSS represents an effective hybridization of two disparate innovation models, "private investment" and "collective action," describing that FOSS incentive structures lead toward free revealing of software developed for private purposes because even after such free revealing the private benefits remain, regardless of whether free riders use the code; moreover, free riders do not take the same value as the originator who gained skills and problem-solving satisfaction in writing the code, and free riders may enhance the revealers' private value by increasing the user base for the code).

10. In using the term "commercial FOSS," I specifically mean to juxtapose the divergent concepts behind free software as compared to the aims of intellectual-property-enabled commercial technology companies.

11. A distinguishing feature of the FOSS movement in the last decade, its time of greatest prominence, is the paucity of court cases in comparison to the movement's increasing importance in information technology. In spite of this, 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,

remained, it would have undermined a foundational premise of FOSS licenses. By determining that copyright remedies were not available to Jacobsen under the FOSS license at issue,[13] the district court diminished FOSS licensing's clever use of copyright to promote, among other things, transparent source code. One of the knowledge production benefits of the FOSS movement has been to inspire open approaches in other areas, including its inspiration for the Creative Commons. The district court decision also detracted from the design of the Creative Commons licenses, showing that court determinations enhancing or diminishing FOSS licensing can also have second-order effects given the reach and influence of the FOSS movement.[14]

On one side of the dualism is the free software strand within the FOSS movement, while on the other is the open source strand. Each correlates to different licensing models and to different practices to gather satisfaction from writing and supplying software. The free software strand would typically use licenses with a mechanism known as copyleft to ensure that the original license conditions (often requiring source code availability and sometimes prohibiting ongoing royalties) remain in place for downstream versions of the software. With this, appropriating value from the software is biased toward services and other economic complements whenever the FOSS developer needs value to accrue to her in a pecuniary fashion.

The Collaborative Integrity of Open-Source Software, 2004 UTAH L. REV. 563; Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265, 268, 274–75 (2004) (discussing FOSS as a social movement: "the legal system must have a framework with which to judge the social value of free software's open development model"). 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) [hereinafter OPEN SOURCES], *available at* http://oreilly.com/catalog/opensources/book/toc.html; *see also* CHRISTOPHER M. KELTY, TWO BITS: THE CULTURAL SIGNIFICANCE OF FREE SOFTWARE (2008), *available at* http://twobits.net/read; OPEN SOURCES 2.0: THE CONTINUING EVOLUTION (Chris DiBona et al. eds., 2006) [hereinafter OPEN SOURCES 2.0]; PERSPECTIVES ON FREE AND OPEN SOURCE SOFTWARE (Joseph Feller et al. eds., 2005), *available at* http://mitpress.mit.edu/books/chapters/0262562278.pdf; 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.*, RON GOLDMAN & RICHARD P. GABRIEL, INNOVATION HAPPENS ELSEWHERE: OPEN SOURCE AS BUSINESS STRATEGY (2005); VAN LINDBERG, INTELLECTUAL PROPERTY AND OPEN SOURCE: A PRACTICAL GUIDE TO PROTECTING CODE (2008); HEATHER J. MEEKER, THE OPEN SOURCE ALTERNATIVE: UNDERSTANDING RISKS AND LEVERAGING OPPORTUNITIES (2008); LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW 103–06, 126–28, 133–36 (2005) (discussing the way in which FOSS licensing developed and how it works), *available at* http://www.rosenlaw.com/oslbook.htm; ANDREW M. ST. LAURENT, UNDERSTANDING OPEN SOURCE AND FREE SOFTWARE LICENSING (2004), *available at* http://oreilly.com/catalog/osfreesoft/book.

12. *Jacobsen I*, No. C 06-01905, 2007 WL 2358628 (N.D. Cal. Aug. 17, 2007).

13. *Id.* at *6–7.

14. ERIC VON HIPPEL, DEMOCRATIZING INNOVATION 99 (2005), *available at* http://web.mit.edu/evhippel/www/democ1.htm ("Open source software has emerged as a major cultural and economic phenomenon.").

Jacobsen's story is the narrative of the stylized FOSS developer who codes and shares for nonpecuniary satisfactions. Jacobsen's group did not use a copyleft license, but many similarly situated groups do so. For historical reasons developers often choose the Free Software Foundation's (FSF) General Public License (GPL),[15] which is a strong copyleft license locking the software under its scope into a development mode characterized by source code availability and a prohibition against ongoing royalties to run the software.

MetaCarta's narrative is that of open source software development within a for-profit company. It applies an attribution-only license to the projects it stewards, meaning that others can deploy or use the software however they wish so long as such later deployment gives attribution to the software's originators.[16] Open source developers sometimes start projects under an attribution-only license to allow for the future involvement of a company under a proprietary or hybridized model.[17] The attribution-only license allows for the possibility to later release the software under either the GPL or as proprietary software, or perhaps as both in a dual-licensing strategy. MetaCarta has the twist of involving itself with patents. Other commercial FOSS entities, however, use kindred mechanisms, such as dual licensing, to rig an appropriability mix that allows some benefits of FOSS development to contribute to the prospects of the entity. Hopefully, as a result, the entity is therefore also a better (more financially viable) steward for the FOSS projects.

FOSS's influences on knowledge production and distribution, the theme for this Symposium, must be considered in light of the free software/open source dualism, but also in light of appropriability.[18] The weight of the

15. References to the GPL will be general in many instances. It is, however, important to note that GPL version 3 (GPLv3), released in 2007, substantially revised version 2 of the license (GPLv2), which was released in 1991. Among the changes most relevant to this article are the provisions handling patent licensing. GPLv2 did not explicitly handle granting and terminating permissions to practice software patent rights. This, along with the need for various other changes, resulted in GPLv3. *See* Free Software Found., GPLv3 First Discussion Draft Rationale (Jan. 16, 2006) [hereinafter GPLv3 First Discussion Draft Rationale], *available at* http://gplv3.fsf.org/gpl-rationale-2006-01-16.html (discussing the decision to create version 3 of the GPL); GNU General Public License, Version 3, § 11 (June 29, 2007) [hereinafter GPLv3], *available at* http://www.gnu.org/licenses/gpl-3.0.html; GNU General Public License, Version 2 (June 1991) [hereinafter GPLv2], *available at* http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

16. 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.

17. Michal Tsur & Shay David, A License to Kill (Innovation)? Open Source Licenses and Their Implications for Innovation 22–23 (Apr. 30, 2005), *available at* http://ssrn.com/abstract=858104.

18. As used in this essay, the term "appropriability" refers to mechanisms to obtain economic value from some effort or investment. *See generally* Joel West, *Does Appropriability Enable or Retard Open Innovation?*, *in* OPEN INNOVATION: RESEARCHING A NEW PARADIGM 109, 109 (Henry Chesbrough et al. eds., 2006), *available at*

literature to date treats FOSS as a nonmarket, peer-production method of developing and distributing new knowledge. FOSS has generated new knowledge in the sense of new collaboration models for software development and market deployment; inspired other movements, such as Creative Commons or free culture generally; and it provides or supports numerous technology platforms, including important elements of the Internet's past and future development.

This impressive scorecard of knowledge production is bronzed by FOSS benefits in knowledge distribution. Simply put, FOSS created a sea change in the availability of source code to study and learn coding and software technology at every level of complexity and in an incredibly diverse array of languages and information technology environments. In other ways, however, the benefits of FOSS are less clear. Superior code quality, in terms of lower defects and greater resistance to problems, is often argued to be a FOSS benefit for structural reasons. Empirical evidence on the point, however, is mixed, although many high-profile FOSS projects are clearly of very high quality. A reframed question is more to the point: is the quality of software developed with the methods of the FOSS movement of higher quality compared to traditional proprietary software development? If so, this is a part of FOSS's contribution to knowledge creation for the information technology ecosystem.

Software is of greater benefit not only if its quality is high, but also if it provides superior functionality. Often superior functionality means new functionality; that is, technology innovation from some programmatic processing, presentation, or interfacing that is novel and heretofore not in existence within information technology. The creation of new nonplatform software functionality may not yet be a primary strength of FOSS development. Assuming this is true, it raises a knowledge production question for FOSS: can the movement gain momentum in generating new nonplatform functionality as opposed to primarily moving functionality from one platform to another, or commoditizing existing software products? Is the mechanism to gain this momentum in the nonpecuniary satisfaction of volunteer developers coupled with the leveraging of economic complements under the free software approach? Or, is the path in open source appropriability with commercial FOSS experiments such as MetaCarta?

These questions are not in a vacuum because other new appropriability mechanisms for software have mainstreamed in the last decade. Thus, the traditional models, such as the proprietary software product vendor model and the custom software developer model, now compete with advertising-supported software and web-delivered software as a service.

---

http://www.openinnovation.net/Book/NewParadigm/index.html (noting that "[f]ormal appropriability by and large depends on intellectual property (IP) laws, and certain types of Open Innovation are only possible through such IP protection").

To consider the potential issues that may come to courts with commercial FOSS that hybridizes proprietary and open source appropriability mechanisms, these other changes in software delivery must be considered. In this essay, it is not possible to consider every imaginable issue that may alter appropriability, so I focus on example issues related to patent law for several reasons.[19] First, for all forms of software, patent law is becoming both a more important appropriability mechanism as well as a greater threat to freedom to operate for developers and users.[20] Second, more FOSS licenses are including or adding provisions to deal with patents held by persons or entities involved with the software. Finally, patent law may be particularly threatening to FOSS due to asymmetries compared to other software development models. These include the low likelihood that FOSS projects will themselves be able to acquire patents defensively, and that source code availability of FOSS allows a proprietary software plaintiff to evaluate infringement easily while a reverse evaluation would be more difficult.[21] Patent law has already influenced FOSS licenses that have provisions that try to clear patent rights in a project. It can support hybridized structures like the MetaCarta example. Thus, it is a likely future influencer of organizational structure, resource flows, and entanglement

19. Besides the reasons given in the main text to focus on patent law, two additional reasons are (1) there has been some treatment of copyright-related FOSS issues in the legal literature; and (2) patent law is structurally in opposition to FOSS. To elaborate on the second point, when a change occurs in copyright law that heightens the power of a copyright holder, that change also cedes more potential to the FOSS movement to govern communities through licenses. The opposite dynamic occurs with patent law. The greater the patent power, the greater potential for a third-party patent holder to disrupt a FOSS community.

20. With respect to patent rights, the phrase "freedom to operate" means the process of clearing patents out of the way for a new technology, to the extent possible. This often means licensing known patent rights held by others and sometimes means searching issued patents for those that might need to be licensed. The process is inherently imperfect for several reasons. First, many issued patents might not be valid if challenged in litigation, but it may be costly to invalidate a patent. Second, the searching process cannot be guaranteed as complete, both due to the number of issued patents and the pliability of patent claim language. Additionally, there may be third-party patent applications in queue at the U.S. Patent and Trademark Office (USPTO) but not yet made public. *See generally* Kenneth W. Dam, *The Economic Underpinnings of Patent Law*, 23 J. LEGAL STUD. 247, 247 (1994) (describing that the patent system is designed to solve the "appropriability problem"—that a firm could not recover the costs of invention if the resulting information were available to all, in which case we could expect a much lower and perhaps suboptimal level of innovation).

21. *See* Stephen M. McJohn, *The Paradoxes of Free Software*, 9 GEO. MASON L. REV. 25, 50–52 (2000) (arguing that systemically, and over the long run, FOSS may inhibit patent infringement suits because open and available source code increases the chance that litigants will discover patent-invalidating prior art); *see also* MEEKER, *supra* note 11, at 93–98 (describing considerations for and against the proposition that FOSS is particularly susceptible to patent litigation, noting issues such as the fact that often volunteer-centric FOSS projects cannot fund a defense, but that often the best "defense" available to the FOSS community is public outcry against a company that wields patents against FOSS; the mere possibility of such outcry is acknowledged to have an effect on decisions taken by companies in these affairs).

within and between FOSS communities and commercial entities that seek to leverage FOSS.

To proceed with these themes, Part I describes the FOSS licensing system. Part II elaborates on the scorecard for FOSS in knowledge production and distribution. Against this background and for transition to the next part, Part III discusses the new appropriability mechanisms shaping information technology. Next, after describing the evolving interface between proprietary information technology companies and FOSS, which includes bidirectional benefits and influences, Part IV argues that even when deployed commercially with appropriation strategies akin to those of proprietary software, FOSS should be recognized by courts and policymakers[22] for its unique public benefit spillovers. This recognition should inform any analysis by courts of the provisions in FOSS licenses, in particular patent provisions that might undermine the efficacy of beneficial FOSS licensing practices within the FOSS movement or within commercial use of FOSS. The essay concludes by emphasizing the implications for knowledge production and distribution at the interface of commercial information technology and FOSS.

I. FOSS LICENSING AS A FOUNDATION FOR KNOWLEDGE PRODUCTION

In Yochai Benkler's treatment in the *Wealth of Networks*, FOSS is a "quintessential instance of commons-based peer production."[23] As an organizing influence for knowledge production of new software, Benkler's approach focuses on the GPL license and its copyleft characteristics, particularly ones channeling the software into a development and distribution mode foreclosing proprietary appropriation.[24] His insight and model show how nonmarket motivations, such as social connectedness via cooperative creative activity, allow information production without commercial value-appropriation techniques.[25] The phenomena Benkler models, however, sometimes operates effectively on software using non-GPL licenses, such as attribution-only licenses, that are permissive

22. An example of a nonjudicial setting where FOSS policy considerations might be important is a federal agency. For example, the Federal Communications Commission (FCC) had to consider issues related to FOSS in wireless devices that use the spectrum that the FCC regulates. *See* SOFTWARE FREEDOM LAW CTR., FCC RULES ON FOSS AND SOFTWARE-DEFINED RADIO 1 (2007), *available at* http://www.softwarefreedom.org/resources/2007/fcc-sdr-whitepaper.pdf (arguing that, "[w]hile the FCC's position on FOSS is more conservative than is necessary, the FCC has given a qualified endorsement to the use of FOSS by [Software Defined Radio] manufacturers by recognizing the benefits of using FOSS in wireless products").

23. YOCHAI BENKLER, THE WEALTH OF NETWORKS: HOW SOCIAL PRODUCTION TRANSFORMS MARKETS AND FREEDOM 63 (2006), *available at* http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf. *See generally* Benkler, *supra* note 11, at 59–69 (postulating formal model describing collaborative peer production for information products, and conditions under which the model will tend toward such peer production, including the necessity of organizing communications among peers, typically through the Internet).

24. *See* BENKLER, *supra* note 23, at 63–68.

25. *See id.* at 4–7.

with respect to future uses of the software. Thus, the licensing lock-in prohibiting privatizing the software is not always a necessary predicate to a successful FOSS project. Sometimes the FOSS community holds together buoyed by the various other factors underlying the peer-production phenomena.

In these instances, a threat to the peer-production ethos of the community is the sense of outrage that may ensue if someone appropriates the project for private gain.[26] The outrage is more likely if that someone is an outsider who has given nothing to the project in the past. The dualism of free software/open source within FOSS helps signal when projects pose that risk for prospective contributing developers. GPL-style free software projects minimize the risk, whereas attribution-only projects admit the problem up front. Given the common, ongoing use of attribution-only licenses, [27] this suggests that some FOSS communities are willing to contribute effort to projects even when others do, can, or might take private value with or without contributing originally or giving back to the community later. This observation helps illuminate the plausibility of hybrid structures, such as for-profit commercial software companies entangled with free software or open source projects.

Among the many indicators of growth and success with the FOSS movement is the proliferation of FOSS licenses. Oftentimes companies or development communities scour the many dozens, if not hundreds, of publicly posted FOSS licenses only to conclude that no license exists that satisfies their preferences. A common next move is to write a new license. Such proliferation may have negative effects on the FOSS movement, to the extent proliferation dissuades standardization and its benefits, or simply makes license selection overly difficult.[28]

---

26. In commenting on this work at the Symposium, Jonathan Zittrain cleverly labeled this effect "the chump factor"—expressing the notion that people disfavor unjust enrichment when they are enriching someone else who is supposed to share, use, or be involved with the FOSS. *See generally* Vetter, *supra* note 11, at 609–14 (discussing the Apache project as an example of open source development under an attribution-only license).

27. MEEKER, *supra* note 11, at 22–26, 43–46 (using the term "permissive license" in the same sense at this essay's use of the term "attribution-only license," noting that permissive licenses are in common use).

28. 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, and dilution of the energy of a group associated with the open source camp that operates a certification mark for FOSS licenses. *See* Open Source Initiative, Report of License Proliferation Committee and Draft FAQ, http://opensource.org/proliferation-report (last visited Mar. 23, 2009). In the Open Source Initiative's (OSI) 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" where "[m]any, but not all, of these licenses fall into the category of vanity licenses." *Id.*

Arti Rai briefly posits a theoretical basis from which one might understand the problems with FOSS license proliferation—an anticommons-type effect. *See* Arti K. Rai, *"Open Source" and Private Ordering: A Commentary on Dusollier*, 82 CHI.-KENT L. REV.

Even with proliferation, the historical success and lineage of, on the one hand, the GPL, and on the other hand, well-known attribution-only licenses such as the Berkeley Software Distribution (BSD) license,[29] allows a rough sorting of FOSS license types. With this understanding of the free software/open source dualism, this part further explains the licensing approaches related to each side.

A. *Copyleft and Foreclosing Appropriation by Privatization*

The free software strand of the FOSS movement prefers licenses that ensure nonprivatization of the software, namely requiring generally available public source code disclosure and prohibiting use royalties. Linked to these goals 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 antiroyalty provision, to intermixed or further-developed software.[30] Embodied in the GPL license, these terms are means to implement a philosophy of functional self-determination and freedom with one's computer.[31] The original implementation was version two of the FSF's GNU General Public License (GPLv2), arriving in 1991.[32] The FSF's progenitor, Richard Stallman, implemented these novel licensing concepts

---

1439, 1441–42 (2007). Rai notes the possibility of a license anticommons arising from "the possibility of conflicting obligations under multiple licenses. This objection may be more theoretical than real, however. In particular areas, there tends to be an informal standard." *Id.*

29. Open Source Initiative, The BSD License: Licensing, http://www.opensource.org/licenses/bsd-license.php (2006) (last visited Mar. 23, 2009) [hereinafter The BSD License]. Several popular FOSS licenses, such as the Apache license, derive from what is known as the Berkeley Software Distribution (BSD) license, used to release the source code of a flavor of the Unix operating system developed at Berkeley. *See* Ruben van Wendel de Joode et al., Protecting the Virtual Commons: Self-Organizing Open Source Communities and Innovative Intellectual Property Regimes 55–56 (Draft Version 1.1, Sept. 2002) (unpublished manuscript), *available at* http://opensource.mit.edu/papers/joode.pdf (discussing pros and cons of practically unrestricted BSD licenses, in particular criticisms from the open source community that it makes no sense because companies "can use, modify and sell the software[,] and the license does not require them to contribute anything back to the community that originally developed the software").

30. 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 11, at 105–06. The Free Software Foundation (FSF), involved in the origination of the label "copyleft," relates it to license-term reciprocity with the purpose of software freedom. *See* GNU Project, What Is Copyleft?, http://www.gnu.org/copyleft/ (last visited Mar. 23, 2009) ("*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) (discussing GPLv2 copyleft).

31. Free Software Foundation, The Free Software Definition, http://www.fsf.org/licensing/essays/free-sw.html (last visited Mar. 23, 2009).

32. GPLv2, *supra* note 15.

in GPLv2 toward his greater ends of software freedom.[33] An ingenious yet incomplete document,[34] 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.

A variety of industry developments in the decades following the GPLv2's arrival combined with the license's potent ideological force and clever use of copyright to propel FOSS licensing into a prominent and pathbreaking place within information technology worldwide. Its force and presence, and lightning rod character, has grown over time, with the GPL[35] remaining the dominant license among its many imitations in mind-share, if not code-share. Even with its success, the GPL paradigm leaves open the debate about how appropriability fits into the motivational mix of software developers. This is due to not only the continuing existence of proprietary software, but also to open source approaches permitting a variety of downstream relicensing models.[36]

B. *Attribution-Only Licenses and Nonforeclosure of Appropriation*

Before GPLv2 in 1991, certain software development communities issued freely redistributable software without strictures dissuading privatization. Among the most famous was a flavor of the Unix operating system from Berkeley under the BSD license.[37] The licensing law theory is straightforward: a short statement claims copyright in the code, but then allows any use on minimal conditions, one of which is that there be attribution to the original code. In other words, the attribution-only license

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

34. 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 GPLv3. *See* GPLv3, *supra* note 15; GPLv3 First Discussion Draft Rationale, *supra* note 15, at 3–4 (discussing the decision to explicitly license patents). *See generally* Greg R. Vetter, *Open Source Licensing and Scattering Opportunism in Software Standards*, 48 B.C. L. REV. 225, 239 n.42 (2007) (analogizing the GPLv3 revision process to a private standard-setting initiative).

35. Occasionally, there may be a need to refer to the GPL without identifying a specific version.

36. *See* MEEKER, *supra* note 11, at 206–07.

37. Marshall Kirk McKusick, *Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable*, *in* OPEN SOURCES, *supra* note 11, at 31, 42–46 (describing the initial decision to offer the entire BSD Unix flavor under the BSD license, due to the popularity of a networking component earlier offered under the license, and discussing a later, related lawsuit that pitted the other major flavor of Unix, from AT&T, against the free BSD Unix distribution, the dispute being whether a small number of copyrighted AT&T components were present in the kernel of BSD Unix); *see also* KELTY, *supra* note 11, at 136–41 (describing how free redistribution of BSD Unix allowed its TCP/IP protocol implementation to become ubiquitous, contributing to the de facto standardization of those protocols for the worldwide Internet).

has a permission set that is easy to satisfy.[38] One need only preserve the notices.

The attribution-only license does not have the features to help 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 incorporation 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 licenses used for FOSS projects.[39] Some of these licenses, despite their straightforward, noncopyleft approach under copyright law, have language to handle patent rights.

Both GPL-style licenses and attribution-only licenses raise a variety of other interesting copyright and contract law issues.[40] However, these issues are put aside in this essay except for the treatment in the next two subsections: clearance of inbound intellectual property rights by contributors and patent provisions.

C. *Inbound Assignments or Licenses by Contributors*

Within active software development communities using the GPLv2 license, an interlocking web of reciprocal copyright license rights ensures that the project remains locked into a nonproprietary development mode. The more dispersed the code contributions, the stronger the lock.[41] On the other hand, if just one developer contributed a dominant or severable major portion of the code, that developer will have greater leverage over the project.

---

38. *See* MEEKER, *supra* note 11, at 43–46, 137 (noting that permissive licenses do not restrain competitors from using the software). The original BSD license was more complicated, but was later simplified. *See* The BSD License, *supra* note 29 (describing that the original BSD license had four clauses).

39. 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.

40. *See generally* Vetter, *supra* note 11, at 644–49.

41. *See id.* at 599–605. The GPL lock-in effect has parallels with the original anticommons theory applied to intellectual property rights, whereby more owners of increasingly small fragments of a resource protected by a system of property rights will lead to underuse of the resource. *See generally* Michael A. Heller & Rebecca S. Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, 280 SCIENCE 698, 698 (1998) (describing how revisions to patent doctrine may create too many fragmented rights and result in too few innovations). In the GPL-type license case, the interlocking rights of many programmers does not lead to underuse within the license conditions, but does preclude use under the proprietary software model where source code is kept secret and users must pay royalties to use.

Some projects seek to eliminate this possibility by requiring the contributors to assign or license their rights to a central organization.[42] David McGowan analyzes the possibility of assignments from the perspective of opportunism:

> One way to combat opportunism is to ask authors of open-source code to assign their rights to an organization controlled by a representative portion of the community. The organization would then decide whether to terminate rights or take code private and, if properly constituted, its decisions might reasonably reflect community sentiment.[43]

With respect to the intellectual property right at issue, typically copyright in the FOSS setting, an assignment will convey all of the copyright interest to the grantee. For an effective assignment regime as part of a FOSS project, contributing developers must trust the grantee organization. A commonly cited example of a trusted organization is the FSF. Due to its focus on enforcement of its FOSS licenses, the FSF prefers assignments over inbound licenses.[44] However, assignments leave no rights in the hands of the contributing programmer. A license, however, allows the programmer to keep some rights.

MetaCarta, the company discussed in the Introduction, uses inbound licenses rather than assignments. Before MetaCarta will accept the contribution to a FOSS project it stewards, the programmer must sign a document licensing copyright interests and patent interests.[45] The copyright license is nonexclusive,[46] thus allowing the programmer to

---

42. Free Software Foundation, Frequently Asked Questions About the GNU Licenses, http://www.fsf.org/licensing/licenses/gpl-faq.html#AssignCopyright (last visited Mar. 23, 2009). The FSF ties its policy of requiring assignments to the possible later need to enforce its FOSS licenses:

> Our lawyers have told us that to be in the best position to enforce the GPL in court against violators, we should keep the copyright status of the program as simple as possible. We do this by asking each contributor to either assign the copyright on his contribution to the FSF, or disclaim copyright on it and thus put it in the public domain.

Id.

43. McGowan, *supra* note 11, at 300. David McGowan further notes that the Free Software Foundation advocates the assignment approach. *Id.* In discussing the implications of a "complete absence of property in the software domain" for open source software, Yochai Benkler notes that "copyright permits free software projects to use licensing to defend themselves from defection." Benkler, *supra* note 11, at 446. He then makes a point similar to McGowan's about the possible value of public mechanisms for preserving open source software: "The same protection from defection might be provided by other means as well, such as creating simple public mechanisms for contributing one's work in a way that makes it unsusceptible to downstream appropriation—a conservancy of sorts." *Id.*

44. MEEKER, *supra* note 11, at 180–81.

45. MetaCarta, The Clear BSD: Introduction, http://labs.metacarta.com/license-explanation.html (last visited Mar. 23, 2009) (describing the terms of the Contributor License Agreement).

46. Specifically, here is the copyright license grant from a contributor to MetaCarta:

> You hereby grant to the Maintainer and to recipients of software distributed by the Maintainer a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare derivative works of, publicly

engage in other licensing activity not conflicting with the grant to MetaCarta, assuming that the code contributed has some potential value apart from the project.

The point of either approach, assignment, or license, is to clear copyright and patent rights in the project. FOSS licenses began to consider provisions to deal with clearing patent rights later in the evolution of the movement. The issues with clearing patent rights are different in degree for various reasons, but the most prominent is that U.S. patent rights allow the holder to exclude anyone in the United States that has issued the patent, regardless of independent development by the alleged infringer.[47] The next section discusses the typical patent provisions in FOSS licenses.

D. *Patent Provisions*

FOSS licenses often employ a standard patent licensing mechanism to specify indirectly the patent(s) covered. Direct specification is often the approach in a traditional patent license. The direct approach lists patent numbers, and perhaps specific claim numbers. The indirect approach refers to the technology with some degree of specificity without mentioning specific patents. The resulting patent license is for patent claims that are embodied by the technology. The indirect technique can vary in a great number of ways.

For example, the indirect approach to licensing a patent claim covering a process of sorting information in a computer might partition the technology by operating system, specifying one set of terms for GNU/Linux,[48] and another set of terms for all other operating systems. If the licensee is active in the market selling its own proprietary software implementing the claimed sorting algorithm, it will want licensor's patent grant to cover at least: (1) relevant specific patents or patent applications held by the licensor; (2) any such future instruments; (3) any such instruments acquired in the future that are embodied by the licensee's current technology; and (4) any

---

> display, publicly perform, sublicense, and distribute Your Contributions and such derivative works.

Id.

47. *See* Sara Boettinger & Dan L. Burk, *Open Source Patenting*, 1 J. INT'L BIOTECH. L. 221, 226 (2004). On some doctrinal implications of increased patenting for software, see generally Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 1, 4–8 (2001).

48. 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, which is a separate open source software effort. *See* Richard Stallman, Linux and the GNU Project, http://www.gnu.org/gnu/linux-and-gnu.html (last visited Mar. 23, 2009). Thus, some use the name "GNU/Linux" for such a distribution. *Id.* The GNU acronym is a self-referential label meaning "GNU's Not UNIX," with Unix being a predecessor computer operating system. *See* GNU Operating System, http://www.gnu.org (last visited Mar. 23, 2009).

such instruments currently owned or acquired in the future that are embodied by the licensee's foreseeable future technology.

Three of these four points deal with future developments, showing one purpose of the traditional patent license as a mechanism to clear a path for present and future commercialization coordinated between the licensor and licensee. This approach can bring some, but not perfect, certainty to patent rights bearing on the technology. To the extent certainty occurs, it is because the licensee is in the best position to foresee the technology trajectory at the time of entering into the license.

Contrast the patent licensee just sketched with a FOSS development team: foreseeability is diminished; perhaps it is gone. Moreover, the licensor is unidentified at the beginning of the project: any future contributor holding patent rights is a potential licensor. These distinctions make a substantial difference. The texture and effects of FOSS patent licensing provisions seem more like the patent issues associated with clearing rights in a standard than many traditional patent-licensing technology agreements.[49]

1. The Patent Grant: Permission to Practice What the Patent Claims as Measured by the FOSS Program

In a typical FOSS patent grant, the provision refers indirectly to the FOSS to which the license document is applied. The FOSS license document, we may recall, also handles copyright in the program. Thus, using version three of the GPL (GPLv3) as an example, one of its conditions is that upon distribution of the software the distributor grants a permission to users of the software to practice patents held by the distributor. "Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version."[50]

---

49. *See* Vetter, *supra* note 34, at 228–29, 238–42.

50. GPLv3, *supra* note 15, § 11. A contributor is "a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's 'contributor version.'" *Id.* A "Program" is "any copyrightable work licensed under this License." *Id.* § 0. Essential patent claims have the following definition:

> A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Id. § 11. While distribution in a traditional copyright sense would qualify, the GPLv3 specifies layered definitions that encompass secondary liability concepts in copyright. *Id.* §§ 2, 8, 11.

A number of authors have analyzed some of the major revisions in GPLv3, including the new patent provisions. *See generally* Stephen J. Davidson & Nathan S. Kumagai,

Thus, there is an important interaction between the copyright-based foundation of the FOSS license, particularly if it seeks to implement "copyleft" for the software, and the patent rights hopefully cleared by the license. In one sense, there is a web of permissions that is reciprocal within copyright and patent, and which crosses between them. But in another sense, because most conditions trigger upon a copyright distribution, the copyright-based conditions are the foundational rights for the FOSS license, particularly because they are always likely to be present. Many FOSS programs will embody no patent claims. But all FOSS programs are concerned with copyright in the code.

Many persons and entities involved with FOSS are both users and distributors. But, often, one status dominates. Thus, a patent-owning technology company that distributes GPLv3-licensed software must account for the possibility that some of its patents will be unavailable for enforcement against users of the FOSS program. On the other hand, a large corporate user of the FOSS program, such as a retailer, that does not principally involve itself with obtaining patents, does not have the same worry. The retailer may use the FOSS as a mission-critical component of its information technology infrastructure. Assume further, however, that the retailer has a subsidiary that develops and licenses proprietary software and also regularly obtains patents on the software. The subsidiary must act prudently in wielding these patents to avoid "retaliation" or "patent peace" provisions in FOSS licenses.

2. The "Patent Peace" Provision: Loss of Patent and Copyright Permissions upon Wielding a Patent

The "retaliation" or "patent peace" provision terminates both copyright rights to use the software, and patent rights in the program in the event of patent litigation of some sort. Thus, using the example above, if the retailer's subsidiary sues a competitor claiming that the competitor's use

---

Developments in the Open Source Community and the Impact of the Release of GPLv3, 938 PRAC. L. INST., PATS., COPYRIGHTS, TRADEMARKS & LITERARY PROP. COURSE HANDBOOK SERIES 537 (2008) (describing industry reactions to GPLv3 along with a short summary of the parent provisions); Robert W. Gomulkiewicz, *A First Look at General Public License 3.0*, 24 COMPUTER & INTERNET LAW. 15, 15, 17–20 (2007) (describing the drafting process for version 3.0 and the structure of the new patent provisions); Robert W. Gomulkiewicz, *General Public License 3.0: Hacking the Free Software Movements Constitution*, 42 HOUS. L. REV. 1015, 1032, 1035–36 (2005) (explaining the way in which the FOSS community interprets the GPL, and arguing generally for a clarification of the language used in newer versions); Sapna Kumar & Olaf Koglin, *GPL Version 3's DRM and Patent Clauses Under German and U.S. Law*, 2 COMP. L. REV. INT'L 33 (2008); Douglas E. Phillips, *Version 3 of the GNU General Public License: Is the Latest Really the Greatest?*, 938 PRAC. L. INST., PATS., COPYRIGHTS, TRADEMARKS & LITERARY PROP. COURSE HANDBOOK SERIES 503 (2008) (discussing the increased complexity of GPLv3 and then providing an in-depth treatment of its DRM, "Tivoization," and patent provisions).

of the FOSS program infringes the subsidiary's patent(s), the subsidiary puts its parent's use of the mission-critical FOSS at risk.[51]

In GPLv3, the retaliation or patent peace provision is as follows: "[Y]ou may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it."[52]

Among other influences, the potency of the patent peace provision increases in response to a larger installed user base, increased functionality, and heightened business criticality of the FOSS program. The more important the software is to a user or distributor, the greater is the anti-patent-wielding preventive effect of the retaliation provision.

The range of effects from these two patent provisions working in tandem depends on the scope of each. Such scope can be expressed in a variety of ways, which is the subject of the next subsection.

3. Interactions and Potential Effects of Patent Grant and Patent Peace Clauses

Two common scope-influencing factors for the patent grant are prominent among the items creating a range of possibilities for its interaction with the FOSS program. One is the time frame for qualifying patent ownership, and the other is the version(s) of the FOSS licensed. Similarly, these same factors influence the scope of the patent peace provision that, when triggered, may limit copyright and patent permissions to use the FOSS program. Moreover, the antipatent emphasis of the FOSS movement hints at greater scope for the retaliation or patent peace provision.

The emphasis in the discussion below is functional and effects-based for patent claim scope and license scope. To describe and demonstrate the effects, it is necessary to put aside challenges that one might imagine to the legal potency of the FOSS licensing approach. The approach has potency generally, but that does not mean that every enumeration below would be effective or valid in whole or in part. Understanding the range of possibilities, I hope, helps the process of analyzing any specific situations that fit within these.

---

51. Another scenario distasteful to the hypothetical retailer is what I call FOSS "tunneling," where the code flows through an organization without corporate understanding. FOSS is attractive to technologists trying to solve problems and operate the information technology infrastructure of a company. An attraction factor is ease of access: the developer finds the code on the Internet, ready for the taking without the trouble of corporate procurement procedures. Among other potential troubles, undocumented and nonenumerated FOSS inflow creates increased risk of inadvertent outflow. The effects of the outflow might change depending on whether or not there were modifications. The issue also raises questions of agency, particularly apparent agency, and corporate action.

52. *See* GPLv3, *supra* note 15, § 10; *see also id.* §§ 0, 2, 8, 10 (propagating or modifying the conveyed "covered work" must be in compliance with the license, which prohibits "further restrictions" and characterizes certain patent assertions as such).

a. *Increasing Scope of Patents Licensed*

A prototypical FOSS patent grant will license any patents owned by the distributor of the software at the time of the distribution, assuming that such distribution obligates the distributor to license its patents. The actual patents licensed are specified indirectly as those with claims embodied by the software distributed. This means that the patent claims might be embodied by the whole program or merely a part of it. The proposition facing the distributing entity seems straightforward: estimate whatever benefit flows from distributing the software against any detriment arising from the loss of the patent enforcement power against users of the FOSS program. Some FOSS licenses, however, open the ownership window to include after-acquired patents, which may change the calculus.

i. After-Acquired Patents

One example of language licensing after-acquired patents can be found in GPLv3, which requires contributors to the FOSS program to license what are defined as "essential patent claims," given as "all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired."[53] Thus, estimating the balance of benefits and detriments is harder given the future uncertainty of patent ownership. The effect is significant, but not catastrophic, to estimating the detriment side because companies involved with the patent system have procedures that regulate selection of technologies for patent applications. The capacity to foresee future patenting is greatest with the distributor, assuming that the distributor undertakes some due diligence and self-evaluation around the FOSS distribution decision.

On the other hand, acquisition of companies or technologies can lead to increased patent ownership. Thus, at some level, the after-acquired addition to the scope of the patent grant represents an opportunity foregone.

It is possible that a FOSS license patent grant would be arranged such that as long as the distributing entity continues to distribute modified software, newly owned patents would be covered under the patent grant even without the after-acquired patents term. This would occur if the mere act of distribution triggered the patent grant. The after-acquired clause thus has the greatest impact when the entity ceases being a distributor. Then, measured in reference to the FOSS program for either the last version distributed, or for future versions as well, future-owned patent claims are also licensed. More precisely, to reorient to the unique structure of FOSS licensing, if the distributing entity wants to remain in compliance with the copyright-rendered conditions linked to its distribution, it will comply with the condition that says it shall license (by not asserting them) patents owned then and owned later.

53. *See id.* § 11.

The approach of GPLv3 is limited to the last version of the FOSS program that the distributor authorized for distribution and use where the distributor was obligated under the patent grant.[54] It does "not include claims that would be infringed only as a consequence of further modification of the contributor version."[55]

ii. Future Versions of the FOSS Program

Although it did not do so, the GPLv3 could have expanded the scope of its patent grant by including future versions of the FOSS program.

A patent grant specifying any future versions of the FOSS program ties the distributing licensor to the trajectory of the FOSS program. The licensor may exert no control, or some influence, over that community, but its ability to cabin or channel the scope of FOSS functionality is less than with proprietary technology. When the patent grant includes future versions, the scope of the permission follows the increasing future functionality, assuming an active, accretive project. Unlike the after-acquired patents term, where the distributor is itself increasing the scope of permission it gave, the future-versions term delegates to the third-party FOSS community involved with the project the indirect specification of patents licensed in the future. The community adds more code, improves the software, and thus increases the chance that it embodies an increasing number of the licensor's patent claims.[56]

When companies involve themselves in FOSS projects, other strategic factors add to the motivational mix. These factors include an entities' estimation of its strategy in light of the patent permission license grant, and whether the grant includes after-acquired patents and/or future versions. These same two factors are the first influencers of scope for the patent peace provision, which also concerns technology companies involved with the patent system and with FOSS.

b. *Increasing Scope of Wielding Patents*

While perspectives on patent protection for software are not favorable within the FOSS movement, the free software camp finds the system particularly venomous. In the United States, the system operates, from a legal perspective, generally without exception from industry to industry, but

54. The definitional taxonomy specifies a contributor to the FOSS program under GPLv3 as follows: "A 'contributor' is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's 'contributor version.'" *Id.*

55. *Id.*

56. This simplified account ignores complications such as whether a fork in the program results in two programs that each must continue to carry the patent license grant. A different perspective on that problem is the question of when a program changes so substantially that it is not the same program even if it carries the same name. The reverse question is also intriguing: What if the program is mostly the same code, but is distributed under a new name?

it impacts each differently from a practical perspective. Information technology companies tend to need portfolios of patents for effective defensive purposes and to buttress competitive advantage both through litigation and licensing for royalties. Interaction with standard-setting is also important because information technology is "layered" both at the hardware level and the software level: protocols and interoperability criteria influence market shares and perceptions. Thus, among many factors that might tempt a patent holder to wield the patent through litigation, involvement with a FOSS program finds a place in the order of strategic concerns.

An interesting issue concerns the pros and cons, particularly with respect to market reputation, of asserting patents against a FOSS community with which one is involved; however, that topic must be put aside to focus on the mechanism(s) that would bring increasing scope to loss of rights to use the FOSS program upon such wielding. Finally, before proceeding to those mechanisms, this discussion intentionally ignores uninvolved patent holders who wield against the FOSS program.[57] This is likely the most common threat to FOSS that has gained commercial prominence, especially given the increase over the last decade in nonpracticing patent holders and contingency fee patent litigation.

i. The FOSS Program Licensed Under the License at Issue

The baseline idea for a patent peace provision is that assertion of one's patents against users of the version one distributed results in a loss of the copyright and patent rights granted by others in that version. How this interacts with the patent grant depends on that clause's scope. If it is only for patents currently owned, then a patent-wielder who stopped distributing but still uses may primarily worry about the patent peace provision as an inhibition to suing FOSS users on newly acquired patents.[58]

When the potential patent-wielder continues to distribute new versions as they come along, the patent peace provision might seem unnecessary because the distributor may be granting the patent license upon each distribution. Even in that situation, however, the patent peace provision helpfully buttresses the license potency due to the often indeterminate mode of enforcement that might accompany use of the FOSS license against the patent-wielder. This refers to a question that needs evaluation in the context of the particular transaction at issue, but is generally the question whether the FOSS license is enforced only under copyright, or also under contract. If nothing else, a specific recitation that the distributor-as-user

57. *See, e.g.*, Complaint at 2–3, IP Innovation LLC v. Red Hat Inc., No. 2:07-cv-447 (E.D. Tex. Oct. 9, 2007), *available at* http://docs.justia.com/cases/federal/district-courts/texas/txedce/2:2007cv00447/105833/1/0.pdf (alleging infringement of three patents by the GNU/Linux operating system as distributed by Red Hat and Novell).

58. As another theoretical wrinkle in the scope of the patent peace provision, consider decoupling it from a distribution and tying it to internal, private use, considering separately modified code versus unmodified operation of the FOSS received.

loses rights to use based on assertion of a patent clarifies the intent of the license drafters.

Thus, the potency and necessity of the patent peace provision will depend on the scope of the patent grant and other factors. This leaves the question of what types of assertions trigger the loss of right to use. That scope does not necessarily have to be tied to the FOSS version distributed by the potential patent-wielder. As with the patent grant, it might be linked to any future version as well. Or, it might go further.

ii. Any FOSS?

The next hypothetical step for a patent peace provision is to extend the peace to all FOSS. For example, envision a provision implementing the following: any assertion of any patent against any FOSS results in a loss of right to use the FOSS supplied under this license.

By omission, this covers currently owned and future-acquired patents. It does not specify whether the condition (written as an obligation) applies only upon distribution, or upon mere internal use, which further shows the range of possibilities. Functionally, the patent-holder must forego assertion of patents against all FOSS programs that embody the asserted patent claims. FOSS is a minority of the worldwide installed base of software, but it is a nontrivial minority in some markets, and a majority in certain niches. Thus, this is a substantial increase in scope for the patent peace provision.

In addition, the example provision above underspecifies FOSS. Does it include FOSS licensed under attribution-only licenses? Should qualifying FOSS be specified by reference to some external standard, such as that provided by the open source camp's certification organization called the Open Source Initiative (OSI)? This is a group operating a certification mark for the movement, called the Open Source Definition (OSD). The OSD defines criteria against which the OSI evaluates and certifies licenses.[59] Alternatively, perhaps the FOSS license should define in its text what is meant by the FOSS to which the patent peace provision attaches.

iii. Any Software?

Beyond the difficulty of specifying what is meant by FOSS, a patent peace provision could theoretically aim itself at assertion of any patent against any software. Some of the early drafts of GPLv3 contained notions of this sort under a mechanism that allowed the person applying GPLv3 to her software to add specialized terms.[60] A heightened scope for

59. Open Source Initiative, http://www.opensource.org (last visited Mar. 23, 2009) ("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.").

60. GNU General Public License Discussion Draft 1 of Version 3, § 7(e) (Jan. 16, 2006), *available at* http://gplv3.fsf.org/comments/gplv3-draft-1.html (allowing modifications to the

the patent peace provision, however, did not survive to the final draft of GPLv3, although the general mechanism of allowing users of the license to customize it, within a menu of allowed options, did survive.[61] Including a greater scope for the patent peace provision on that menu had other disadvantages for GPLv3 not relevant to the present discussion. What is important is the relationship between greater scope in the patent peace clause and clearing patent rights in the FOSS program.

If more patent-holding companies, foundations, or individuals participate in a FOSS development project, this presents a greater potential for clearing a greater number of patent rights in the software. How high this reaches depends on scope provisions for the patent grant, such as after-acquired patents and the future-versions term. But if patent-holders who would like to be involved with the FOSS shy away due to an overly broad patent peace provision, this would be counterproductive. These dynamics practically limit the scope of such clauses, putting aside any potential legal doctrines that also might diminish their efficacy.

The next logical step in the progression of this subsection is assertion of any patent covering anything as a trigger to loss of one's right to use the FOSS. I am not aware of any FOSS license that attempts this. The free software camp of the FOSS movement finds the patent system unjust as applied to software, and perhaps unjust generally, but practical concerns make this last step seem implausible.

One result of patent provisions in FOSS licenses is increasing complexity. For example, GPLv3 substantially increased its complexity compared to GPLv2.[62] While the core ideas of free software are relatively straightforward, the use of intellectual property law to implement those "freedoms"[63] is circuitous, particularly with the addition of patent law in the licensing structure. An empirical question that will be impossible to ever answer is whether GPLv2 achieved some of its success because it did not have patent law provisions until many years after some other FOSS licenses, and thus had reduced complexity. Whatever the answer to that inquiry, some perspectives are clear as to the success of the FOSS movement generally. This is the topic of the next part.

II. SUCCESSES AND ASPIRATIONS FOR FOSS KNOWLEDGE PRODUCTION

Stepping away from the gory details of FOSS licensing terms, this part catalogs the observed successes and aspirations of FOSS as a knowledge production paradigm. While this scorecard process is itself interesting, a

license to impose conditions triggered by the assertion of a software patent lawsuit); GPLv3 First Discussion Draft Rationale, *supra* note 15, § 2.12.11.

61. GPLv3, *supra* note 15, § 7.

62. *See* Phillips, *supra* note 50, at 507–12.

63. GNU Project, The Free Software Definition, http://www.gnu.org/ philosophy/free-sw.html (last visited Mar. 23, 2009).

proper introduction suggests acknowledging that something fundamental is going on. As Christopher M. Kelty puts it,

> Free Software exemplifies [a cultural] reorientation [of knowledge and power]; it is not simply a technical pursuit but also the creation of a "public," a collective that asserts itself as a check on other constituted forms of power—like states, the church, and corporations—but which remains independent of these domains of power. . . . Free Software is a public of a particular kind: a recursive public. Recursive publics are publics concerned with the ability to build, control, modify, and maintain the infrastructure that allows them to come into being in the first place and which, in turn, constitutes their everyday practical commitments and the identities of the participants as creative and autonomous individuals.[64]

Except for the first item in the list, the topics below focus on knowledge production as the creation of new platform and nonplatform software. My focus is at once both more narrow and more broad than Kelty's conception of free software as a recursive public. It is more narrow because the list gives some of the important knowledge outputs from FOSS,[65] but does not tie them to the greater cultural significance of Kelty's free software conception. It is more broad because I am interested in nonpublic (often nonplatform) software production apart from the platform/public software that constitutes supporting layers for the recursive public that is free software.[66]

A. *Inspiring New Ecologies for Collaboration and Information Sharing*

In its rise as a major influence on the information technology ecology, FOSS attracted notice from other domains. Some within these domains have imported FOSS techniques and adapted them to other modes of collaborative sharing and infrastructure development.[67]

Undoubtedly the most prominent example of this phenomenon is the Creative Commons project.[68] What FOSS licenses are to the proprietary

---

64. KELTY, *supra* note 11, at 7 (footnote omitted).

65. In addition to the direct benefits of each of the items on the FOSS scorecard in this part, there may be second-order effects of greater productivity or heightened learning about computing from any of these items.

66. *See* KELTY, *supra* note 11, at 3–6 (introducing the concept of a recursive public and its relating effect among itself, free software, and the Internet).

67. Christopher M. Kelty describes this process as "modulating" the practices underlying free software into other domains. *See id.* at 245–47 (describing two collaborative projects, Connexions and Creative Commons, as modulations of free software practices); *see also* Connexions, http://cnx.org/aboutus/ (last visited Mar. 23, 2009) ("Connexions is an environment for collaboratively developing, freely sharing, and rapidly publishing scholarly content on the Web.").

68. *See* Creative Commons, http://creativecommons.org/ (last visited Mar. 23, 2009) ("Creative Commons provides free tools that let authors, scientists, artists, and educators easily mark their creative work with the freedoms they want it to carry."); *see also* Lydia Pallas Loren, *Building a Reliable Semicommons of Creative Works: Enforcement of Creative Commons Licenses and Limited Abandonment of Copyright*, 14 GEO. MASON L. REV. 271, 278 (2007) (arguing that courts should "draw on the copyright doctrine of

software world, the Creative Commons licenses are to the commercial world of music, movies, publications, and other media.[69] The Creative Commons approach is more centralized than the FOSS movement in that a central organization coordinates a standardized menu of licenses.[70] The licenses allow an author or creator to select the type of sharing allowed for her work. With the benefit of the FOSS experience before it, the Creative Commons movement has grown rapidly. Internationalization of the licenses along with establishment of affiliate organizations in many other countries follow FOSS's forerunner success.[71]

Modulating FOSS practices into other domains includes the domains of biotechnology[72] and scientific publishing.[73] In addition, Benkler catalogs various other collective activities, including the prominent example of Wikipedia, that fit within his peer-production model and arose contemporaneously with the emergence of FOSS in the public consciousness.[74] That FOSS inspires new ecologies underscores its far-reaching nature and influence. Within the information technology ecology, a part of that influence has been to establish platforms.

---

abandonment to create a new doctrinal category of limited abandonment" for general use and to support the Creative Commons licenses).

69. *See* Severine Dusollier, *Open Source and Copyleft: Authorship Reconsidered?*, 26 COLUM. J.L. & ARTS 281, 282–87 (2003); Niva Elkin-Koren, *What Contracts Cannot Do: The Limits of Private Ordering in Facilitating a Creative Commons*, 74 FORDHAM L. REV. 375, 387–88 (2005).

70. *See* Creative Commons, About: Licenses, http://creativecommons.org/about/licenses (last visited Mar. 23, 2009). In FOSS, the license-generation process is somewhat ad hoc, with certain important licenses such as GPLv2 garnering imitators, but without a centralized authority to design and promulgate the only licenses in use within the movement. On the other hand, the influence of the OSI with its certification mark process, and the influence of the FSF in orchestrating a deliberative procedure for GPL revision in version 3, shows that FOSS license generation is not completely ad hoc.

71. *See* Creative Commons, International, http://creativecommons.org/international/ (last visited Mar. 23, 2009) ("Creative Commons International (CCi) works to 'port' the core Creative Commons Licenses to different copyright legislations around the world. The porting process involves both linguistically translating the licenses and legally adapting them to particular jurisdictions."); *see also* Severine Dusollier, *Sharing Access to Intellectual Property Through Private Ordering*, 82 CHI.-KENT L. REV. 1391, 1394–96, 1400–02 (2007).

72. *See* Robin Feldman, *The Open Source Biotechnology Movement: Is It Patent Misuse?*, 6 MINN. J. L. SCI. & TECH. 117, 118 (2004) ("Building on the software notion of 'copyleft,' some open source biotechnology projects use the power of the patent system to ensure that the core technology of the project and any innovations remain openly available."); Sapna Kumar & Arti Rai, *Synthetic Biology: The Intellectual Property Puzzle*, 85 TEX. L. REV. 1745, 1763 (2007) ("The idea of a synthetic biology commons draws inspiration, in part, from the prominence of the open-source software model as an alternative to proprietary software.").

73. *See* Dusollier, *supra* note 71, at 1405–07.

74. BENKLER, *supra* note 23, at 68–81.

B. *New Information Technology Platforms*

Information technology platforms are synonymous with FOSS.[75] This includes specific projects, such as the Linux kernel,[76] but also includes FOSS influence over de facto standards such as the TCP/IP protocols that underlie the Internet.[77] A variety of considerations influence whether a specific FOSS project has platform potential, beginning with whether it is aimed at a platform market.[78] An important consideration among these, however, is the freely redistributable nature of FOSS.[79] Besides the obvious attractiveness of a subsidized production input, the no-royalty nature of GPL-style FOSS reduces transaction costs on the front end of software acquisition.

Success with platform applications is clearly part of the story of FOSS's success.[80] Beyond the Linux kernel, examples include the Apache web server software,[81] and the Firefox browser.[82] The success in platform applications is consistent with both Benkler's conception of free software as peer production[83] and Kelty's conception of it as a recursive public.[84] In both cases, the platform success of a particular application amplifies its role in the production of more or other FOSS, and in the production of a more egalitarian information technology infrastructure.[85]

---

75. *See* Douglas Lichtman, *Property Rights in Emerging Platform Technologies*, 29 J. LEGAL STUD. 615, 615 & n.1 (2000) (describing a "platform" as something a consumer can purchase "to enhance the value of some number of independently purchased goods," and offering operating systems as an example of a platform technology, while describing application software that runs on the platform operating system as "peripherals" to that platform).

76. Linux Online!, http://www.linux.org/ (last visited Mar. 23, 2009).

77. KELTY, *supra* note 11, at 139–41.

78. Vetter, *supra* note 30, at 126 n.186.

79. *See* Nicholas Economides & Evangelos Katsamakas, *Linux vs. Windows: A Comparison of Application and Platform Innovation Incentives for Open Source and Proprietary Software Platforms*, *in* THE ECONOMICS OF OPEN SOURCE SOFTWARE DEVELOPMENT, *supra* note 9, at 207, 207–09; Joel West, *How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies*, 32 RES. POL'Y 1259, 1259–66 (2003) (discussing how the emergence of standardized platforms which allow for substitution of "complementary assets" has been a driving force for the evolution of the computer industry).

80. Greg R. Vetter, *Exit and Voice in Free and Open Source Software Licensing: Moderating the Rein over Software Users*, 85 OR. L. REV. 183, 256–63 (2006).

81. Apache HTTP Server Project, http://httpd.apache.org/ (last visited Mar. 23, 2009) ("Apache has been the most popular web server on the Internet since April 1996.").

82. Mozilla, Firefox Web Browser, http://www.mozilla.com/en-US/firefox/ (last visited Mar. 23, 2009).

83. BENKLER, *supra* note 23, at 320–23, 394–96.

84. KELTY, *supra* note 11, at 13–18 (summarizing the five practices by which the book *Two Bits* characterizes free software development as a recursive public).

85. Another factor channeling FOSS to platform applications is that many FOSS licenses desire to comply with the Open Source Definition, a certification mark with requirements that licenses do not discriminate in particular ways, allowing FOSS under those licenses a wide applicability. *See* Open Source Initiative, The Open Source Definition §§ 5–6, 10, *available at* http://opensource.org/docs/osd (last visited Mar. 23, 2009) (prohibiting

C. *New Collaboration Modes for Software Development*

FOSS created a new paradigm for distributed, Internet-enabled collaboration to develop software. As the preceding section shows, the paradigm was so successful it found its way into other domains. Benkler's theoretical work links these areas with a descriptive account as to when peer production, such as FOSS, might take hold.[86] The peer-production model depends in part on easy communications among contributors. The Internet typically meets this need, but each domain will need collaboration tools adapted to its work.

For software, these tools are a new generation of source code control systems. This is software to track the changes to the software, but also to transparently show who did what to which piece of code when, and with what authority.[87] The source code control tools evolved to handle social interaction and technical argumentation with respect to the code, making them a portal for tacit knowledge and upending conventional wisdom that developers needed to be in the same physical location for effective sharing of tacit knowledge.[88]

Besides the evolving tools to manage a particular FOSS project, the ecology developed repositories. The most well-known is SourceForge.[89]

---

discrimination against persons or groups, or against fields of endeavor, and requiring technology neutrality, if license is to meet OSD certification).

86. *See* Benkler, *supra* note 11, at 434–36.

87. Vetter, *supra* note 11, at 626–30 & n.184.

88. Sverre Helge Bolstad, Learning and Knowledge in FLOSS: Situated Learning and Organizational Knowledge-Conversion in Community-Based Free/Libre Open Source Software Development 8 (2006) (unpublished Master's thesis, University of Bergen, Norway), *available at* http://opensource.mit.edu/papers/Learning-and-knowledge-in-FLOSS.pdf (arguing "that FLOSS, as an online community of practice, . . . overcome[s] the problem of tacit knowledge-transformation through dense interaction, collective reflection and accumulation of knowledge"); Andrea Hemetsberger & Christian Reinhardt, Sharing and Creating Knowledge in Open-Source Communities: The Case of KDE 4 (2004) (unpublished manuscript), *available at* http://opensource.mit.edu/papers/hemreinh.pdf. Andrea Hemetsberger & Christian Reinhardt describe their research into the workings of a FOSS development community as geared toward answering three questions: "how community members organize content with regard to their daily routines that potentially transforms into knowledge for other members"; "how new members are enabled to accumulate the knowledge necessary for becoming a valued member"; and "how members co-create and conceptualize new ideas—create new knowledge—in absence of physical proximity."

Examples of two popular and complementary FOSS development tools are Trac and Subversion. *See* Trac, Welcome to the Trac Open Source Project, http://trac.edgewall.org/ (last visited Mar. 23, 2009) ("Trac is an enhanced wiki and issue tracking system for software development projects. . . . It provides an interface to Subversion . . . ."); Tigris.org, Subversion, http://subversion.tigris.org/ (last visited Mar. 23, 2009) ("Subversion is an open source version control system."). Between the two tools, a FOSS project can seamlessly integrate the details of: code development; finding, tracking, and fixing defects; tracing all such activity through the source code with full information about which developers/users did what; using Wiki-type and instant messaging communications among developers; and other capabilities.

89. *See* SourceForge.net, http://sourceforge.net (last visited Mar. 23, 2009); *see also* SourceForge.net, What Is SourceForge.net?, http://apps.sourceforge.net/trac/sitedocs/wiki/

The repositories are an important part of the new collaboration modes for FOSS because they enable software searching and discovery of projects as a user or as a potential developer. They also enable developers to easily establish projects with contributing developers worldwide.[90] The number of projects and users registered at SourceForge is impressive, indicating vitality in the FOSS movement even if most of the projects are small and inactive.[91]

Consensus maintains that the first three items in this part's scorecard of FOSS are successes of the movement. The remaining two items, however, are the subject of some debate.

D. *Higher-Quality Code*

Higher-quality software is an often-cited advantage of the FOSS movement.[92] There are important structural reasons suggesting why FOSS might on average produce higher-quality code than proprietary software. These structural reasons relate to the availability of source code and the transparency of design this engenders, coupled with assumptions about the behaviors of an active developer group and user community.[93] There are

---

What%20is%20SourceForge.net? (last visited Mar. 23, 2009) [hereinafter What Is SourceForge.net?] ("SourceForge.net is the world's largest open source software development web site. We provide free services that help people build cool stuff and share it with a global audience.").

90. SourceForge.net is also a unique research asset for studying various aspects of the FOSS phenomenon. *See* J.-M. Dalle et al., Advancing Economic Research on the Free and Open Source Software Mode of Production 9–11 (Dec. 3, 2004) (unpublished manuscript), *available at* http://opensource.mit.edu/papers/davidetal.pdf.

91. What Is SourceForge.net?, *supra* note 89 ("As of February, 2009, more than 230,000 software projects have been registered to use our services by more than 2 million registered users . . . .").

92. Jean-Michel Dalle & Paul M. David, *The Allocation of Software Development Resources in 'Open Source' Production Mode* 2 (Stanford Inst. for Econ. Policy Research (SIEPR) Discussion Paper No. 02-27, 2003), *available at* http://129.3.20.41/eps/io/papers/0502/0502011.pdf. A predicate question is the definition of quality for software, but for purposes of this article I proceed based on a common sense of the word, although formal definitions exist. *See* Martin Michlmayr, Quality Improvement in Volunteer Free and Open Source Software Projects: Exploring the Impact of Release Management 9–13 (Mar. 2007) (unpublished Ph.D. dissertation, University of Cambridge), *available at* http://opensource.mit.edu/papers/michlmayr-phd.pdf.

93. The first word on this topic belongs to Eric Raymond. He has characterized factors, including issues of reliability, that might drive one to choose FOSS development over proprietary development. ERIC S. RAYMOND, *The Magic Cauldron*, *in* THE CATHEDRAL & THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY 143 (rev. ed. 2001) (arguing that criteria to choose between open source or traditional approach includes that one "can expect that open source has a high payoff where (a) reliability/stability/scalability are critical, and (b) correctness of design and implementation is not readily verified by means other than independent peer review," and noting that "[t]he second criterion is met in practice by most non-trivial programs"). Raymond also coined the famous phrase supporting the argument that massive peer review of source-code-transparent FOSS projects would facilitate quality: "Given enough eyeballs, all bugs are shallow." ERIC S. RAYMOND, *The Cathedral and the Bazaar*, *in* THE CATHEDRAL & THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY, *supra*, at 19, 30.

structural counterarguments, which include the possibility of free riding by one user or developer waiting for another to fix bugs, although a counterpoint can be made that such free riding will not manifest when user needs are heterogeneous and this is known among users.[94] These and other aspects of the debate are hard to test empirically in a conclusive fashion, particularly since there are both high-quality and low-quality software products in both the FOSS and proprietary worlds.[95]

Certain FOSS products are recognized to be of very high quality, such as the Linux kernel or the Apache web server.[96] The ultimate question, which cannot be answered here, and in some sense can never be answered, is whether the FOSS approach scales to high quality all around, or greater quality on average than proprietary software.[97] If this question engenders debate, then the last section in this part will further the discussion because its metric, innovativeness, is an even more expansive and diffuse concept than quality.

---

94. Michlmayr, *supra* note 92, at 14–22 (collecting sources and summarizing the literature on FOSS quality assurance practices and comparative quality with proprietary software). Martin Michlmayr concludes his FOSS quality literature review with the following:

> In summary, the FOSS methodology incorporates a number of elements, such as potentially high levels of peer review and user innovation, that may contribute to quality. In recent times, a body of academic research on FOSS has been established which gives supports to anecdotal claims that FOSS can attain high levels of quality, in particular regarding code quality. However, there is also increasing awareness that the FOSS methodology is facing a number of challenges [such as usability, contributor burnout and resource limitations], some of which are unique to the collaborative development effort performed by volunteer participants. Given that this area of research is still very young and that much focus has been on technical matters . . . further investigation is necessary.

Id. at 21.

95. For an example of the sometimes contentious nature of the debate, a series of postings on Slashdot is instructive. *See* Ben Chelf, *Insecurity in Open Source: What Open-Source Developers Can Learn About Security and Quality from—Gasp—Makers of Proprietary Software*, BUSINESSWEEK.COM, Oct. 6, 2006, http://www.businessweek.com/technology/content/oct2006/tc20061006_394140.htm?campaign_id=bier_tco.g3a.rss1007 (based on a study undertaken by the company Coverity, the study authors argue that FOSS can improve quality by adopting methods of testing from some proprietary software products, but they note that "on average, open-source software is of higher quality than proprietary software"); *see also Bug Hunting Open-Source vs. Proprietary Software*, SLASHDOT, Oct. 7, 2006, http://linux.slashdot.org/article.pl?sid=06/10/07/173255 (giving a variety of reactions to the Coverity study).

96. GOLDMAN & GABRIEL, *supra* note 11, at 47–48. The authors also argue that more early releases under a FOSS model allow for a greater reduction of defects as compared to the proprietary software model. *Id.* at 85–86.

97. Although it is beyond the scope of this essay, the comparative quality question can be reframed with an expanded conception of quality that considers data security issues such as malware and spam. *See generally* Peter P. Swire, *A Theory of Disclosure for Security and Competitive Reasons: Open Source, Proprietary Software, and Government Systems*, 42 HOUS. L. REV. 1333 (2006) (discussing the pros and cons of secrecy versus disclosure in systems software from a security perspective).

E. *New Nonplatform Software Functionality*

The more one accepts Kelty's conception of free software as a recursive public, the more one can believe in the FOSS movement as a driver for new platform software. But an additional inference that follows is that there may then be less motive force for nonplatform or nonpublic FOSS software. That debate relates to the topic of this section. Like the preceding section, comparative quality of FOSS versus proprietary software, the "new functionality" question is one with unknowable answers in an empirical sense.[98] For purposes of this section, however, I am assuming that the answer is affirmative for generating new platform functionality. This leaves the question regarding nonplatform functionality.

Anecdotal accounts vary on the issue, as do perspectives on how to sort the question. Enterprise software used internally by companies, such as accounting and operational software, is an area that many feel has little FOSS presence, although those applications often depend on platform software, such as databases, that might or might not be FOSS.[99] Assuming as correct the characterization that FOSS has minimal presence in these areas, there are many nonexclusive static explanations about why.[100] In a

98. By use of the term "new functionality," I am leaving some concepts fluid, such as the extent to which software functionality has to differ from what came before to be "new." There is also the concept of new to a platform. For example, there was a point in history where personal computers did not have a windows interface, but Macintosh computers did. Another distinction is between invention and innovation. *See* Rebecca S. Eisenberg, *Patents and the Progress of Science: Exclusive Rights and Experimental Use*, 56 U. CHI. L. REV. 1017, 1036–37 (1989) (distinguishing the act of invention from the follow-on act of innovation, where the latter involves activities to commercialize the former).

99. Charles Babcock, *The Open Source Enterprise—The Problems that Made Open Source Code Impractical for Many Businesses Are Falling Away. Combined with the 'Cheaper' Factor, This Should Get Interesting*, INFO. WK. (N.Y.), Nov. 17, 2008, at 41, *available at* 2008 WLNR 21938519 ("Open source code still can't touch the scope of proprietary suites, but it's closing the technology gap. . . . With its transparent and standards-based development, open source code can cut the complexity and risk of custom coding for integration or niche needs."); *Open Source Databases Widely Used, Seldom Paid for: Open Source Systems Have Replaced Only a Few Commercial Products in the Enterprise*, COMMWEBNEWS.COM, Apr. 1, 2008, *available at* 2008 WLNR 6172514 (describing a report that "concludes that the expected impact of open source code to commoditize the database market has yet to occur").

100. A first explanation for a purported minimal presence of FOSS in enterprise applications is the head start of the proprietary vendors—they have been providing accounting and operational software to businesses for many decades. A second, related explanation is vendor lock-in due to switching costs. *See, e.g.*, Charles Ferguson, *How Linux Could Overthrow Microsoft: The Open-Source Movement Is the Largest Threat the Software Giant Has Ever Faced. Does Bill Gates Have a Plan?*, TECH. REV. (Mass.), June 1, 2005, at 64, 69, *available at* 2005 WLNR 8789992 (relating how vendor "lock-in" describes the disincentives a company has to switch to an alternative technology, which include switching costs and network effects of the installed technology). A third explanation might be that FOSS products do not yet offer equivalent functionality compared to proprietary vendors in these application spaces. A fourth, related explanation is that FOSS products, even when providing more or better functionality, do not provide a better value perception for the prospective product-switching user due to perceptions of negatives with FOSS, such as lack of centralized support or diminished usability.

dynamic sense, those explanations may dissolve over time as companies investigate greater use of FOSS for either its cost advantages or for new functionality it may eventually provide. This is the crux of the issue: when can the lack of FOSS penetration in a software market be attributed to insufficient improvements in functionality to nudge a user into a product switch? When will the FOSS improvements need to create functional parity with the proprietary product, and when will they need to outpace?

Another perspective on the issue is to consider temporal structure. FOSS evolved and emerged along with the Internet, so many of its platform functionality advances are associated with the Internet. Many other types of important software functionality, however, predated the Internet. Thus, business applications were already entrenched in accounting, operations, design, financial transactions, and other areas. This leads to the inference that, over time, reimplementations of these applications may appear as FOSS in a process that seeks to commoditize these application areas.[101] If this occurs, the winners are users in the application space, and the losers are the proprietary software vendors currently serving those users, assuming that the FOSS implementations provide a better value proposition.

Since its emergence, FOSS has generated benefit in the nonplatform space in the sense of learning spillovers due to the availability of its source code. Thus, programmers working on an internal, custom business application might examine FOSS code at a repository such as SourceForge.net to learn how to program a particular method. This type of transference is an important part of the FOSS ecology. It also underscores the reality that much internal business application software is not from a software product provider.[102]

Related to the question of learning spillovers from FOSS is the question of software functionality that requires significant research and development (R&D) funds and a long gestation period in the R&D pipeline before commercial deployment. This could include core, embeddable technologies such as encryption or speech recognition, or domain specific applications, such as design software used in the electrical or structural engineering fields. Sometimes, the R&D and scientific development in these areas use open science or open source approaches,[103] but some organizations will continue to operate assuming that certain types of investments require appropriability mechanisms beyond the complementary goods/services

101. *See, e.g.*, OFBiz, The Apache Open for Business Project, http://ofbiz.apache.org/ (last visited Mar. 23, 2009) ("The Apache Open For Business Project is an open source enterprise automation software project licensed under the Apache License Version 2.0. . . . Apache OFBiz is a foundation and starting point for enterprise solutions . . . ."); Posting of Dan Farber to ZDNet, http://blogs.zdnet.com/BTL/?p=1502&part=rss&tag=feed&subj=zdblog (June 14, 2005, 2:55 PM).

102. James Bessen, *What Good Is Free Software?*, *in* GOVERNMENT POLICY TOWARD OPEN SOURCE SOFTWARE 12, 18–22 (Robert W. Hahn ed., 2002), *available at* http://aei-brookings.org/admin/pdffiles/phpJ6.pdf.

103. Carnegie Mellon University, Speech at CMU, http://www.speech.cs.cmu.edu/ (last visited Mar. 23, 2009) (showing some software research outputs available as open source).

recoupment available under GPL-type FOSS licenses. For example, Apple might fit this mold. It continues to rely in part on conventional proprietary model exclusivity and related appropriation mechanisms.

There is some empirical research that attempts to characterize the degree of innovativeness of FOSS applications. One study of interest evaluated active projects on the SourceForge.net repository through a human reading and coding of the project descriptions: it found that most projects commoditized functionality from proprietary software products or ported functionality to a new operating system.[104] As the study acknowledges, the focus on SourceForge projects creates a sampling problem,[105] but the results are consistent with anecdotal observations I have fielded, and fit with the temporal structure argument discussed above.

The extent to which FOSS applications in the nonplatform space will eventually surpass, in some grand sort of average assessment, the functionality of preexisting proprietary applications depends on many factors. Some of those influences are the subject of the next part. New modes to appropriate value from software may promote or suppress transparency of source code and software development, depending on both external factors and on considerations internal to the FOSS movement and the ways in which it progresses.

III. NEW MODES OF APPROPRIABILITY FROM SOFTWARE

Apart from FOSS itself, there are two other Internet-enabled software appropriation mechanisms gaining prominence. They are related in that

---

104. *See generally* Krzysztof Klincewicz, Innovativeness of Open Source Software Projects (Aug. 11, 2005) (unpublished manuscript), *available at* http://opensource.mit.edu/papers/klincewicz.pdf (in a review of the five hundred most active projects on the SourceForge.net FOSS repository web site, finding minimal innovation as assessed from the project descriptions; and finding most projects are porting functionality to a new operating system platform, or replicating in FOSS the functionality from the proprietary software environment). *But see* Christof Ebert, *Open Source Drives Innovation*, IEEE SOFTWARE, May/June 2007, at 105, *available at* http://www2.computer.org/portal/web/csdl/doi/10.1109/MS.2007.83; Dario Lorenzi & Cristina Rossi, Innovativeness of Software Solutions: Evidence from an Alternative Methodology: Comparing Free/Open Source and Proprietary Products 1, 16, 18 (n.d.) (unpublished manuscript), *available at* http://opensource.mit.edu/papers/lorenzi_rossi_MIT_20071220.pdf (focusing on small and medium sized software companies in Italy, attempting to determine whether "programs based on FOSS [are] more innovative than proprietary ones," with the "sample of 134 software solutions . . . formed by 109 proprietary and 207 FOSS solutions," finding "that proprietary and FOSS software not only show different levels of innovativity, but, as far as, *new to the world products* are concerned, they are also shaped by different innovation processes: radical innovation in the FOSS vs. incremental innovation in proprietary field").

Neither of the two studies discussed in this footnote differentiate the objects of the study as platform versus nonplatform, thus providing no support for my suggestion that new functionality will be less likely to come from FOSS in the nonplatform application space as compared to the platform space. That inference remains supported only by structural arguments. *See* Vetter, *supra* note 80, at 256–62.

105. Klincewicz, *supra* note 104, at 13 ("Even though SourceForge is the most representative collection of OSS projects, it does not cover all relevant open source communities.").

each depends on external computing. Thus, advertising-supported software depends on the Internet for a value proposition whereby third-party advertisements reach a user who enjoys the software capabilities in exchange for bearing the advertisements. Software as a service is the Internet era's version of a long-standing business model in computing: provide metered-for-a-price computing capabilities. Each of these finds the external computers somewhere in the "cloud" out on the Internet. The term "cloud" has become a popular phrase to characterize the type of virtual, Internet-delivered computing capabilities increasingly in use by everyone in their devices, phones, and computers.[106]

This part reviews both of these cloud-based modes of appropriability and also reviews FOSS under a conception of "copyleft capitalism"—an extension in degree of the complementary economic models driving much FOSS development.[107] Along with the two new cloud-based modes, FOSS is a new mode of appropriability because a decade ago most information technology observers doubted the ability of companies to establish and sustain revenue models giving away software.

A. *Advertising-Supported Software*

Google's search engine is synonymous with financially successful advertising-supported software.[108] Its success in that area allows it to cross-subsidize a number of other popular software services, such as Gmail or YouTube.[109] Most advertising-supported software is delivered over or using the Internet in a "software as a service" delivery mode. The payment model is different from the application examples in the next subsection. Google is paid by advertisers who want a link to appear for their products or services in a prominent place with the listings that Google's Internet

106. Steve Lohr, *A New Battle Is Beginning in Branding for the Web*, N.Y. TIMES, Aug. 31, 2008, *available at* http://www.nytimes.com/2008/09/01/technology/01copyright.html?th=&emc=th&pagewanted=all ("Cloud computing usually refers to internet services or software that the user accesses through a Web browser on a personal computer, cellphone or other device. The digital service is delivered remotely, from somewhere off in the computing cloud, in the fashion of Google's internet search service.").

107. There are a number of potential business models applicable to FOSS. *See* MARTIN FINK, THE BUSINESS AND ECONOMICS OF LINUX AND OPEN SOURCE 175–89 (2003). These include using open source software to enable hardware and/or service sales, the core of IBM's strategy. The list also includes dual licensing, and service and support such as that provided by Red Hat, and others. *Id.*

108. Google, Investor Relations, http://investor.google.com/faq.html#money (last visited Mar. 23, 2009) (noting that "the majority of [Google's] revenue comes from advertising").

109. Google, More Google Products, http://www.google.com/intl/en/options/ (last visited Mar. 23, 2009); *see also* Posting of Josh to eCommerce and Entrepreneurship Blog, http://www.plumbersurplus.com/Blog/post/2008/12/Life-in-the-Cloud-Beginning-the-Journey-with-Google-Apps.aspx (Dec. 12, 2008) ("[W]e're testing a transitioning of our users over from the clunky and resource intensive (and expen$ive) Microsoft Outlook, Word, Excel, Powerpoint, MSN Live Messenger, etc over to Google's cloud model of Google Apps, Gmailesque email, Google Docs, Google Talk, and Google Sites. So, this week, I will be 'living in the cloud' and completing 100% of my work on email, documents, spreadsheets, and presentations from Google Apps.").

search capability provides. It, along with other search engines, continually evolves the search algorithms for various goals, including better targeting for the advertiser to her desired audience.[110]

The example of Google shows that advertising-supported software is viable in at least some instances. There are numerous questions about scope, scale, and the degree of necessary network effects for viability. We need not delve into those questions for the point needed in this essay—that advertising revenues have arisen as an important revenue appropriation technique for certain classes of platform software, such as search and some types of content. Within this application space, some examples of less glamorous products exist.[111] But the plausibility of the model seems proven by Google, with FOSS consequences that resonate with source code transparency along with the software-as-a-service model. Moreover, Google's success is fueled by the flagship FOSS project, GNU/Linux.[112] Thus, since the cloud is the Internet, FOSS underlies the cloud.

B. *Software as a Service*

Just as Google's advertising-supported service is cost-subsidized by the availability of a free operating system upon which to create their search infrastructure, so are other software subscription services subsidized by freely available FOSS.[113] The irony of this situation is that, even under most GPL-style licenses, the copyright works in the user interface that may be delivered to the user from the cloud to the user's device are not the source code literary works covered by the FOSS license. In other words, the interaction of a remote user with the software residing somewhere in the cloud is not a distribution of the source code because typically the information technology design does not make the source code available to the remote user.[114] This gives the service provider the opportunity to

110. *See* Eric Goldman, *Search Engine Bias and the Demise of Search Engine Utopianism*, 8 YALE J.L. & TECH. 188, 189 (2006) (noting that "search engines have significant power to shape searcher behavior and perceptions").

111. 1-800 Contacts, Inc. v. WhenU.com, Inc., 414 F.3d 400, 404–05 (2d Cir. 2005); *see also* Press Release, WhenU, WhenU Wins Landmark Internet Decision (June 28, 2005), *available at* http://www.whenu.com/press_release_05_06_28.html.

112. Google is reported to run a modified Linux-kernel-based operating system, but it does not make the source code for its modifications available. *See Google's Summer of Code Pays Students to Do Open Source*, DATAMONITOR, June 9, 2005, *available at* 2005 WLNR 9127797; Interview by Jason Schumaker, Linux Journal, with Sergey Brin, Co-Founder and President, Google, Inc. (Sept. 1, 2000), *available at* http://www.linuxjournal.com/article/4196.

113. SugarCRM, SugarCRM Defines the New CRM Generation, http://www.sugarcrm.com/crm/products/crm-products.html (last visited Mar. 23, 2009) (describing how the company's customer relationship management software can be deployed using a subscription, on-demand pricing model, and also describing how the software is available as FOSS and the company's usage of and commitment to FOSS concepts in a commercial, for-profit context).

114. *See* Vetter, *supra* note 80, at 269–72 (discussing the incentive pros and cons bearing on the possibility of source code disclosure in association with actions by the software over the Internet that do not rise to the level of a copyright distribution); *see also* Atl. Recording Corp. v. Howell, 554 F. Supp. 2d 976, 981–84 (D. Ariz. 2008) (in the context of online

directly monetize the freely available FOSS via the service, without contributing to the community supporting the FOSS. What percentage of service providers free ride FOSS in this way is impossible to know, but some clearly recognize that in the long term their situation is improved if they are involved positively with the FOSS community.[115]

Of course, a software-as-a-service provider might not use FOSS. The revenue model is metered computing capabilities, saving the end user the costs of investing in on-site servers and application software.[116] How the service provider provisions internally is a function of many factors. For example, the applications it wants to offer might not be available as FOSS, or on FOSS platforms such as the GNU/Linux operating system. For the user, there are numerous pros and cons of the traditional, in-house information technology approach versus meeting some or all of its corporate software application needs from the cloud as a service. Cost factors, experience and expertise, data control and security, and mission-criticalness will all influence the user's choice.[117]

Just as advertising-supported software providers are similar to software-as-a-service providers through their use of cloud computing, both face a similar choice when they use FOSS: to what extent do they involve themselves in the community? This choice, however, also extends to companies using FOSS internally without offering cloud computing services to the world. The effect of these choices in aggregate is a consideration in the next section.

C. *Copyleft Capitalism and Spillovers*

In a 2008 interview, free software luminary Professor Eben Moglen discussed the need for businesses, as much as individuals, to exercise autonomy over their information technology.[118] The mode for achieving this autonomy is forming a critical mass that treats software as a renewable, commons-like resource.

---

music sharing, finding that merely making files available for downloading did not constitute a distribution when evidence was lacking that the files were actually downloaded).

115. *See, e.g.*, SugarCRM, *supra* note 113.

116. Many decades ago, before the era of personal computers and the Internet, selling time on a remotely located mainframe computer was called "time sharing." In the current era, the software-as-a-service model is also sometimes called the application service provider (ASP) model. The two phrases are synonyms for our purposes.

117. These choices sometimes exist for personal software as well. For example, Intuit's popular Quicken personal money manager has always been available as software to run on a personal computer so the user's data is with her locally. Recently, the company offered the product under a software-as-a-service model. *See* Quicken, Quicken Online—Free, Easy Money Management, http://quicken.intuit.com/online-banking-finances.jsp?lid=site_banner (last visited Mar. 23, 2009).

118. Todd Weiss, *Q&A: Open-Source Backer Eben Moglen Says Software a 'Renewable' Resource*, COMPUTERWORLD, Dec. 3, 2007, http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyId=17&articleId=9050379&intsrc=hm_topic (excerpting from interview with Eben Moglen, Founder of the Software Freedom Law Center).

> One of the things that everybody now understands is that you can treat software as a renewable, natural resource. . . . like forest products or fish in the sea. If you build community, if you make broadly accessible the ability to create, then you can use your limited resources not on the creation or maintenance of anything, but on the editing of that which is already created elsewhere. . . .
>
> . . . .
>
> . . . If you've become dependent on a commons for whatever role in your business, then what you need is commons management. You don't strip mine the forest, you don't fish every fish out of the sea. And, in particular, you become interested in conservation and equality. You want the fish to remain in the sea and you don't want anybody else overfishing. So you get interested in how the fisheries are protected.[119]

One example of a company seeming to follow this approach is SugarCRM. Its software is used by customer representatives to manage relations with customers—thus the acronym "CRM" for customer relationship management. CRM software is a classic internal enterprise business application. It might allow an employee to follow sales leads, issue quotes, track support concerns, manage contacts and communications, and perform other common interfacing activities. Companies traditionally either developed their own CRM application, or procured one from a proprietary software vendor. SugarCRM seeks to penetrate this market with an ideology hoping to seed "copyleft capitalism." Their software can be deployed either as a service or onsite in one of two onsite delivery modes.[120] Critically for SugarCRM's approach, they "author and release Sugar Open Source to a community of 2400 CRM experts and developers who use the software, provide feedback and develop extensions to the Sugar."[121] After initially publishing their own FOSS license, they recently adopted GPLv3, further hoping to promote critical mass.[122]

Companies like SugarCRM are evolving the FOSS complementary goods and services business model thus far associated with companies like IBM and Red Hat. There is an intentional design in the SugarCRM approach to spin up critical mass. In other words, they want a sufficiently large network of users/contributors in order to approximate the renewable commons story Professor Moglen relates. The copyleft commons motif is an extension of the thus far prevalent appropriability mechanisms for FOSS. A key success-threshold question in a submarket of all enterprise software, such as the CRM submarket, is how much of the market must leave the proprietary approach on either the user or provider side? Interorganizational learning behavior might accelerate reaching critical mass in a particular submarket

119. *See id.*
120. SugarCRM, *supra* note 113.
121. *Id.*
122. SugarCRM, SugarCRM GPL v3 FAQs, http://www.sugarcrm.com/crm/gplv3-faq.html (last visited Mar. 23, 2009).

once other submarkets are fully involved in copyleft capitalism. Thus, the critical mass question has importance in a broad market sense for all of information technology, both for platform and nonplatform applications.

Important theorists from the business literature and the legal literature provide two accounts to couple around "copyleft capitalism." The connecting element among these accounts is that many persons or organizations are in a position when involved in FOSS to act as both a user and a contributor. Some persons stake out a place as one or the other, but those positions could change in the future. In the business literature, Eric von Hippel discusses user innovation communities, both generally and specifically for FOSS, cataloging the balance of incentives that often lead innovators to freely reveal innovations.[123] A part of the balance is the possibility that adopting third parties will add value reclaimable by the originator. In the legal literature, Mark Lemley and Brett Frischmann challenge the conventional account that property rule systems, such as intellectual property in the form of copyright and patent law, should exclusively focus on internalizing externalities, arguing that, often, due to demand-side information conditions in a dynamic analysis, externalities express as positive spillovers that a property rules regime may want to encourage.[124] The insight is particularly apt for intangibles such as software protected by intellectual property due to the inherent uncertainty of rights in such.[125] Further, FOSS licensing, particularly under copyleft, fits within the concatenation of von Hippel's work with that of Lemley and Frischmann.

---

123. *See* VON HIPPEL, *supra* note 14, at 77–78, 81–87 (noting that the revealer has asset-specific information, meaning that adopting users have take-up costs, describing the practical difficulties with appropriating value from a kept-secret innovation due to perfect or close substitutes existing with others who may reveal, and articulating the incentives to freely reveal, including reputation, increased diffusion leading to informal standardization, and the possibility that third-party improvements will lower the originator's cost structure to use the innovation).

124. *See* Frischmann & Lemley, *supra* note 9, at 258, 274–84 (noting in the introduction that "in some cases, spillovers actually drive further innovation"); *see also* Katherine J. Strandburg, *Users as Innovators: Implications for Patent Doctrine*, 79 U. COLO. L. REV. 467, 470, 473–74 (2008) (noting that the "picture of user innovation is in sharp contrast to the picture of innovation that dominates discussions of patent policy in the legal literature" and evaluating a proposal to exempt nonsales research use from patent infringement); Jonathan Barnett, *Sharing in the Shadow of Property: Rational Cooperation in Innovation Markets* 58 (Univ. S. Cal. Ctr. in Law, Econ. & Org. Research Paper No. C08-22, 2008), *available at* http://law.usc.edu/academics/centers/cleo/working-papers/cleo/documents/C08_22_paper.pdf (noting that "the open-source model exhibits much of the characteristics of a sharing regime insofar as it generates a common innovation pool in the form of unprotected code").

125. Demand-side information is often unknowable in a precise way to support formation of pricing preferences, leading to underdemand compared to what would be socially optimal for an innovation. *See* Frischmann & Lemley, *supra* note 9, at 263–64, 279–80 ("Unlike a reduction in supply associated with the monopoly deadweight loss problem, the problem here is that demand is reduced, in the sense that the demand manifested by productive users falls short of social demand.").

Fitting within the motif of copyleft capitalism, FOSS under GPL-type treatment reduces demand-side (user-side) valuation issues for ongoing use of code because the license implements a semicommons whereby value extraction comes primarily from complements or downstream reuse of inputs by others. With respect to foreseeable complements, such as service or support associated with a FOSS product, these complements are often easier to value at greater certainty both on the user and supplier side, in part because such valuation does not depend upon vendors estimating the following items: long-term network effects (potentially enhanced with FOSS due to its interoperability advantages); user uptake; or aggregate functionality across all contributed code components. This last valuation decision is made on the supply side piece by piece as contributors decide whether it is worthwhile to freely reveal their innovations by adding to the code.[126] Moreover, the mere existence of a FOSS ecology reduces the supply-side cost to make a project available. A user need only post the project on SourceForge.net and she instantly has the infrastructure needed to promote and evolve the project. Thus, characteristics of the ecology enabled by FOSS further facilitate growth in the movement.[127]

Within this recursive effect, copyleft capitalism in software operates against a baseline. Many decades of proprietary software characterize an industry that has grown to become a major component of any developed economy. That industry, information technology, becomes more diffuse as it grows, creating opportunities for new modes such as FOSS, and situations where FOSS itself will become entangled in situations of mixed incentives and diverse appropriability opportunities.

IV. COMMERCIAL FOSS AND APPROPRIABILITY

FOSS is heterogeneous. The dualism of free software/open source is a simplification that may effectively model the past, but will be increasingly blurred in the future. Commercial interests are becoming more involved with FOSS, resulting in what are called "vendor-driven" projects. In contrast, "community-driven" projects are volunteer-centric, but some of the "volunteers" may be paid by their respective employers to participate in the project.[128] A primary point of differentiation with a vendor-driven project is that no single company dominates the community. The evolution of vendor driven projects, including their increasingly sophisticated use of licensing and intellectual property in strategic concert with other non-IP mechanisms of appropriability, is one concern of this Part. The other is

126. *See* VON HIPPEL, *supra* note 14, at 81 (noting that "the real choice facing user innovators often is whether to voluntarily freely reveal or to arrive at the same end state, perhaps with a bit of a lag, via involuntary spillovers," where involuntary spillovers refer to the practical difficulty of keeping innovations secret, or the practical reality that often others have created, and will reveal, the same or similar innovation).

127. KELTY, *supra* note 11, at 27–30.

128. Linus Dahlander & Martin W. Wallin, *A Man on the Inside: Unlocking Communities as Complementary Assets*, 35 RES. POL'Y 1243, 1247 (2006).

how such uses reflect on the FOSS movement generally, and on courts' perspectives about the movement when the time comes for the next case with critical importance such as that seen with *Jacobsen v. Katzer*.[129]

Toward these concerns, before Part IV.B examines hybridization of proprietary software and FOSS, the first section in this part describes the beneficial spillovers that FOSS generates for proprietary software.

A. *FOSS Benefits for Proprietary Software*

Judging from some of the rhetoric that has echoed from some proprietary software companies, nothing about FOSS could in any way be good for information technology or proprietary software.[130] Reflecting the tensions bearing on proprietary software, Microsoft officials are reported to have hesitated in choosing whether they would rather see their software products "pirated" than have users install and run FOSS applications.[131] Additionally, FOSS has been called a "destroyer" of intellectual property.[132] Most of this rhetoric is for strategic purposes, particularly in relation to governmental and corporate information technology procurement. And certainly there are structural puzzles about what sustains FOSS, particularly the volunteer-centric projects, but questioning its sustainability does not necessarily repudiate its current beneficial effects.

The discussion in Parts II and III above demonstrates some of the beneficial spillovers from FOSS for proprietary software, particularly as to knowledge production and distribution. Learning opportunities from studying source code are a benefit because available source is bounteous due to FOSS. The repositories of FOSS code are not static as a learning tool. If a learner follows a project over time, she can learn from the electronically logged technological discussions of technical merit that guide software architecture decisions. She can test the waters with attempts to contribute as well, provided that doing so is not prohibited by her employer.[133] These learning benefits are available to all, not just to FOSS programmers.

Another benefit is adaptation of FOSS collaborative development practices and software tools for community-style development outside of mainstream FOSS. For example, some companies established closed community software development approaches adapted from FOSS

129. 535 F.3d 1373 (Fed. Cir. 2008).

130. Joseph Scott Miller, *Allchin's Folly: Exploding Some Myths About Open Source Software*, 20 CARDOZO ARTS & ENT. L.J. 491, 491–93 (2002) (describing the rhetoric and counterrhetoric around a Microsoft executive's statement that open source software is an "intellectual-property destroyer").

131. *Piracy vs. Open Source Choice Stumps Microsoft Executive*, WARREN'S WASH. INTERNET DAILY, July 18, 2003 (on file with author) (in debate with Professor Lawrence Lessig, a Microsoft executive is reported to have strategically avoided the question whether the company would prefer developing countries to use open source software or pirated proprietary software from Microsoft).

132. Miller, *supra* note 130, at 492–93.

133. *See* LINDBERG, *supra* note 11, at 181, 193–96.

practices.[134] Some collaborative software development tools (software products/projects themselves) have been deployed internally by major information technology companies. For example, the most well-known FOSS repository, SourceForge.net, itself is available as a software development collaboration tool for businesses under an enterprise license.[135]

In addition, the platforms enabled by FOSS benefit proprietary software. For example, while the Apache web server takes market share from Microsoft's web server product, it generally benefits many non-FOSS software installations, whether they are proprietary software or internal corporate networks. Harm to one particular proprietary competitor does not necessarily translate into aggregate harm for proprietary software generally, particularly for platforms. The general premise of volunteer-generated FOSS seems, in the abstract, a threat to any proprietary software product, but whether the threat is real in a particular market depends on many factors.[136]

Moreover, FOSS sometimes extends the network value of proprietary software. For example, Microsoft's server operating system benefits from interoperability with GNU/Linux servers as a result of a FOSS project that provides interoperability of important system administration data.[137] With source code transparency, FOSS has innate advantages for interoperability. Greater interoperability among information technology components typically enhances the value of any particular component,[138] although in a dynamic process certain types of interoperability are seen as a competitive threat. For example, compatible file formats reduce switching costs for users to change word processing software, a threat to the software provider with greater market share. An extension of these points is that the

134. GOLDMAN & GABRIEL, *supra* note 11, at 67–71.

135. CollabNet, CollabNet SourceForge Enterprise, http://www.collab.net/products/sfee (last visited Mar. 23, 2009).

136. Greg R. Vetter, *Slouching Toward Open Innovation: Free and Open Source Software (FOSS) for Electronic Health Information*, 30 WASH. U. J.L. & POL'Y (forthcoming 2009) (manuscript at 42–51), *available at* http://www.law.uh.edu/faculty/gvetter/documents/VetterSlouchingTowardOpenInnovation-FOSSforEHI_6.29.2008.pdf) (describing a case study of the software product market for software to manage electronic medical records at hospitals or physician offices, developing six factors indicative of whether FOSS uptake is likely in a particular software market).

137. Samba, http://us3.samba.org/samba/ (last visited Mar. 23, 2009) ("Samba is an Open Source/Free Software suite that has, since 1992, provided file and print services to . . . numerous versions of Microsoft Windows operating systems."). The Samba project allows organizations with multiple servers to incorporate servers running GNU/Linux into a Microsoft Windows-based computing environment to achieve various feats of interoperability, such as allowing the GNU/Linux servers to have access to domain management information, such as user profiles. *See* Samba, What Is Samba?, http://us3.samba.org/samba/what_is_samba.html (last visited Mar. 23, 2009) ("Samba is a software package that gives network administrators flexibility and freedom in terms of setup, configuration, and choice of systems and equipment.").

138. Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CAL. L. REV. 479, 491–95 (1998).

advantages of FOSS interoperability pressure proprietary software to itself become more interoperable.

FOSS has already beneficially changed the ecosystem of information technology.[139] These changes have both direct and disciplining benefits for proprietary software. Some of those benefits are discussed above in this section, but one other benefit should be mentioned: the ability for proprietary software to directly incorporate attribution-only FOSS into royalty-bearing products. This practice is discussed below because it is one of several practices this essay includes in the category of commercial FOSS. In this category, a software vendor is the dominant force for a project rather than the traditionally conceived volunteer-centric FOSS community.

B. *Hybridizing FOSS and Proprietary Software*

Paralleling community-driven FOSS's inventive use of licensing and other law to organize collaboration, vendor-driven FOSS is also evolving. Licenses such as the GPL preclude certain types of commercialization.[140] For example, companies seeking to build a business by distributing the GNU/Linux operating system,[141] which is mostly licensed under the GPL, typically rely on some form of complementary value extraction, such as selling affiliated goods or services and/or charging for the packaged distribution.[142] On the other hand, at one point in its history, the most well-known GNU/Linux distributor, Red Hat, sold off-the-shelf copies of GNU/Linux in retail channels. The GPL itself allows this one-time, at -he-time-of-sale, distribution charge. The charge makes sense considering not only the cost of the media and packaging, but also the effort required by Red Hat to find, harvest, organize, and arrange in concert the many hundreds of FOSS components necessary to provide a complete operating system package. Considering only copyright licensing terms, what these companies cannot do under the GPL is exclude competitive entry as to

139. The items given in the main text are focused on two ideas: things that specifically help proprietary software, and things that relate to knowledge production. For completeness, but recognizing that it goes beyond that intersection, another FOSS benefit accruing to any type of organization is the potential for reducing cost in information technology operations. *See* OLLIANCE GROUP, 2007 OPEN SOURCE THINK TANK: THE FUTURE OF OPEN SOURCE: EXECUTIVE SUMMARY REPORT 5 (2007), *available at* http://thinktank.olliancegroup.com/ostt2007report.pdf (highlighting a presentation by Tony Perkins, Founder and Editor of Red Herring, where he noted that "[t]he cost of starting an Internet company plummeted by over 80% from 1996 to 2004" and that the "trend was largely enabled by open source software and powerful, cheap hardware").

140. *See* Dahlander & Wallin, *supra* note 128, at 1249.

141. *See supra* note 48.

142. In this context, the word "distribution" does not refer to the distribution right in copyright, but to that word used as a label for a concerted bundle of FOSS components. For example, various firms provide "distributions" of the GNU/Linux operating system. These distributions are packages of many FOSS components.

distribution of the very same software.[143] Once distributed under the GPL, any other person can also distribute the software under a different trademark even if in direct competition with the company that expended the initial effort to harvest and arrange the package.

Vendor-driven projects deployed under an attribution-only license, in contrast to the GPL, may have some degree of exclusionary power when the released FOSS is a component of the vendor's larger product, system, or service offering. In other words, the attribution-only licensed FOSS is supplemented or entangled with additional layers of proprietary software for the vendor's commercial product. The proprietary layers are likely protected by at least the law of trade secrets, trademark, and copyright, and perhaps also by patent law. In this scenario, the following proprietary software, vendor-appropriation mechanisms are in play once a customer/user has been secured: first-mover advantages, either generally in the market or for the specific customer at issue; switching costs for the customer/user coupled with the potential need for ongoing support, maintenance, and new versions, all of which may allow regular monetary extractions from the customer/user to the vendor; and opportunities to sell other products or services when the vendor is diversified.

FOSS has influenced the marketing process of securing a new customer for all forms of software. This leads an increasing number of vendors with varying degrees of seriousness to espouse (or even implement) some type of "support" for open source. This is probably a part of MetaCarta's motivation in its seemingly legitimate support of its FOSS projects. From the user's perspective, there are interoperability and anti-vendor-lock-in advantages in the FOSS narrative, although realizing the advantages requires business process reengineering and is not automatic.[144] Thus, according to the puffery of the sales situation, the more a proprietary software vendor can be seen to embrace FOSS, the more it might be viewed favorably at the time of purchase by a prospective user. If such embracing is a sham, without any real support for a FOSS community or any real FOSS benefits within the vendor's offering, it is a marketing gimmick.

Beyond layering attribution-only FOSS with proprietary software, another vendor licensing technique to explicitly hybridize proprietary software approaches with FOSS approaches is the dual license.[145] The

143. J. T. Smith, *Red Hat: You Can Distribute Red Hat Linux, Just Name It Something Else*, LINUX.COM, Dec. 10, 2001, http://www.linux.com/feature/19797 ("Basically, users are still able to download Red Hat Linux for free and programmers are still able to base their own distributions on Red Hat, but anyone who distributes a non-authorized copy of Red Hat can't call it Red Hat without the company's permission. Call it Fred's Linux or Generic Linux—several distributions including Mandrake have used Red Hat as a base to build their own products . . . .").

144. Ebert, *supra* note 104, at 52.

145. *See* MEEKER, *supra* note 11, at 143–46 (providing a general overview of dual licensing); Robert W. Gomulkiewicz, *Entrepreneurial Open Source Hackers: MySQL and Its Dual Licensing*, 9 COMPUTER L. REV. & TECH. J. 203, 209–11 (2004) (describing dual

most prominent example of a dual-licensed product is probably the database company MySQL. After about a decade in which MySQL developed the software, its developer community, and its user base, Sun Microsystems purchased MySQL in early 2008 for about one billion dollars in consideration.[146]

For MySQL, dual licensing works as follows. Anyone who distributes their software under the GPL can take and use the MySQL database under GPL terms. But if someone, typically a "value added reseller" (VAR), uses proprietary distribution terms, a commercial license is required from MySQL. This typically occurs when the VAR embeds the database into another software product. For effective dual licensing, a vendor such as MySQL needs pristine handling of intellectual property rights for any inbound contributions from the community.

The three appropriation mechanisms reviewed thus far in this section can be labeled as follows: (1) complements, (2) incorporation, and (3) dual licensing; where incorporation is expressly understood to only apply to attribution-only licensed FOSS. Items two and three are direct appropriation, whereas item one, complements, is indirect. Similar to item one in indirectness are the new software appropriation models of advertising support and software as a service (SaaS). The former requires scale for success. The latter is generally applicable within the reach of the broadband Internet: large or small software companies can use the SaaS model wherever they can reach Internet connected users. Both models can and do take advantage of FOSS, with or without providing modifications back to the community. The condition in copyleft licenses requiring disclosure of modifications is typically triggered upon a distribution of the software. Web-delivered SaaS, as well as web-delivered advertising-supported software, is not a distribution because it is merely web-delivered. No source code is typically distributed; only the functionality and the interface travel over the Internet. Thus, echoing how proprietary software benefits from FOSS, these two new software models also benefit from FOSS. The appropriability mechanism for SaaS is to keep the source code secret and couple this with the lock-in effects of housing the user's data. Along with typical software familiarity lock-in effects, such as user retraining to switch, these influences aggregate to perhaps make the SaaS user more locked in than a traditional proprietary software user who at least keeps her own data on her own computers.

The common strand through all of these appropriation mechanisms is that they are predominantly customer-centric as opposed to competitor exclusionary. Incorporation and dual licensing allow exclusion of competitors to a degree, but, without something other than the power of

licensing generally, and describing specifically MySQL's dual-licensing implementation, which included a need to handle license compatibility issues arising from the GPL).

146. Press Release, Sun Microsystems, Sun Microsystems Announces Agreement to Acquire MySQL, Developer of the World's Most Popular Open Source Database (Jan. 16, 2008), *available at* http://www.sun.com/aboutsun/pr/2008–01/sunflash.20080116.1.xml.

copyright, competitors can often implement functionality to compete. In other words, the opportunity to work around a competitor's copyright leads some companies to the power of patents. This brings us to the MetaCarta example and its patent-based approach.[147] Of course, companies can design around patents as well, but the leverage a patent might provide over competitors is often more potent than that of copyright.

Given the availability of the patent system, a competitor exclusionary strategy based on patents, whether fully or partially exclusionary, is an understandable business decision. Whether it is beneficial in a given market or context is always a question. MetaCarta's patent acquisition may have the ultimate goal of excluding competitors while permitting the FOSS communities to practice any patent claims that are embodied by the FOSS projects. Or, MetaCarta may have patents in response to its venture capital investors. Alternatively, its patents may claim systems or methods that are not embodied by the FOSS projects, although even if that is true there is the possibility that some or all of a FOSS project embodies an element of what is claimed. In other words, patents seem rational to exclude competitors, or to satisfy investors, but the claim scope and coverage also influences whether the patent strategy makes sense. For MetaCarta, the logic might be as follows: the more the claim scope does not involve the FOSS projects, except perhaps with the FOSS as a minor element of what is claimed, the more the patent strategy might be thought to cover innovation that needs value appropriation of the type obtainable with patent protection.

In concept, one might label the MetaCarta approach as sustainable coexistence, where the opposite of that concept is a patent holder, such as Microsoft, that some believe wants to disrupt or destroy certain FOSS communities, such as those developing the Linux kernel, or turn the communities into patent-royalty-paying users. Whether a commercial FOSS company, such as MetaCarta, operates in sustainable coexistence depends on whether and how it wields its patents and their claim scope.

Specifically, under a Contributor License Agreement (CLA), MetaCarta requires inbound contributing developers to grant the FOSS project a nonexclusive patent license for identified patents covering the version of the software to which the code contribution attaches.[148] After-acquired patents

147. At the time of this writing, the USPTO has issued one patent to MetaCarta. U.S. Patent No. 7,117,199 (filed Oct. 3, 2006). In addition to this issued patent, thirty applications are published and attributed to MetaCarta as assignee. *See* U.S. Patent and Trademark Office, Patent Application Search, http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnetahtml%2FPTO%2Fsearch-adv.html&r=0&p=1&f=S&l=50&Query=an%2Fmetacarta%0D%0A&d=PG01.

148. MetaCarta, Contributor License Agreement, § 3 [hereinafter CLA], *available at* http://labs.metacarta.com/license-explanation.html. The Contributor License Agreement (CLA) grants a license to the project maintainer and software recipients with the following scope:

> [P]erpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those

are not included in the CLA license grant, and the CLA also includes a patent peace clause.[149]

On the outbound side, for its FOSS projects, MetaCarta generated a modified version of the BSD license, an attribution-only-type license, where it specifically disclaimed any express or implied licenses under any party's patents.[150] This stands in contrast against other BSD-style licenses that include a grant of a patent license as well as a patent peace provision.[151]

The inference from MetaCarta's approach is that it desires to inoculate itself against the law of implied license in patent licensing whereby some potential defendant in a patent infringement suit claims an implied license, pointing to MetaCarta's support for the FOSS projects. If the gambit is successful, MetaCarta would presumably be able to hold the leverage of its patents over any competitors that practice what the patents claim. It might exclude the competitors and supply the market itself or license the competitors. If these competitors are third parties uninvolved with any of the FOSS projects, this looks like the typical competitor exclusionary use of patents in information technology. If those competitors or their employees contribute to any of the three MetaCarta-supported FOSS projects, the situation is more interesting, but still does not parallel Microsoft's disruptive stance toward the GNU/Linux operating system.

From a vendor's perspective, the purpose of hybridized structures like MetaCarta's approach is to add the strategic advantages of patents. These advantages may include direct appropriation such as licensing or selling to customers in place of excluded competitors, or secondary effects such as influence on standard setting; support for price discrimination or product differentiation; signaling to investors or others; and, in the FOSS context, protection of particular projects by defensive use of patents.[152] The most

---

> patent claims licensable by You that are necessarily infringed by Your Contribution(s) alone or by combination of Your Contribution(s) with the Work to which such Contribution(s) was submitted.

Id. The patents covered by this license are specifically identified by number, allowing the contributor to list patents for which it is not granting a license. A catchall provision states that any owned and issued patents at the time of the contribution are deemed licensed if not listed, but, significantly, this provision does not reach to patent applications. *Id.*

149. *Id.* The CLA patent peace clause terminates only patent licenses to the patent suit plaintiff and therefore is significantly less potent than the comparable clause in other FOSS licenses, such as GPLv3. *Compare id.* (terminating patent licenses), *with* GPLv3, *supra* note 15, §§ 8, 10–11 (terminating all rights, including copyright licenses).

150. MetaCarta, Clear BSD License [hereinafter ClearBSD], *available at* http://labs.metacarta.com/license-explanation.html ("NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS LICENSE.").

151. *See* The Apache Software Foundation, Apache License, Version 2.0, § 3, *available at* http://www.apache.org/licenses/LICENSE-2.0 (explicit patent grant and patent peace clause).

152. Nigel Howard et al., *Use Patent Law to Protect Open-Source Software: Supplement the Incomplete Coverage Provided by Copyright and Trade Secret Law*, N.J. L.J. Nov. 12,

prominent example of this last point is IBM. Its hybridized approach to commercial FOSS includes contributing many resources to important platform FOSS projects, complementary sales of services and other information technology, and dedication of some patents to a commons to support FOSS, all the while keeping other groups of patents to support licensing revenues. IBM is not the typical case. Few companies are situated to both dedicate large numbers of patents to help FOSS yet retain even larger numbers in a commercial licensing program.

The need for a patent commons for FOSS extends to community driven projects, and might also benefit commercial FOSS. IBM contributed to this commons, but this commons will likely never include all patents, which underscores that information technology innovates and operates under the influences of the patent system.[153] This observation leads to the last section of this Part to touch upon the current influences of the patent system and examine a few potential issues that may arise in the future.

C. *Commercial FOSS in the Shadow of Patent Law*

For FOSS, patent law remains an ominous dark cloud on the horizon that has influenced FOSS licensing while the FOSS community's voice against patents has influenced the worldwide debate about software patents.[154] The voice against patent law includes the improbable proposal of eliminating what are often called software patents, but also includes more feasible actions to improve patent quality for patents covering computer-implemented inventions.[155] Whether and when the dark cloud will produce a specific FOSS storm is not for this essay to predict. Instead, I use two patent law issues to consider how commercial FOSS, with appropriation mechanisms from patent law, may fare with courts, or other policymakers such as agencies, from the perspective of a knowledge production narrative. Those two issues are implied license and injunctive remedies.

2001, at 25 (discussing various possibilities to use patent law in defense or assistance of FOSS).

153. *See* John R. Allison, Abe Dunn & Ronald J. Mann, *Software Patents, Incumbents, and Entry*, 85 TEX. L. REV. 1579, 1593–97 (2007); Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?*, 20 HARV. J.L. & TECH. 1, 3–4 (2006); Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961, 988 (2005) (noting that certain information technology areas "had markedly higher [patenting] rates, including graphics and digital imaging, expert systems and natural language, multimedia, and security").

154. Vetter, *supra* note 80, at 248–56.

155. *See* Peer to Patent, http://www.peertopatent.org (last visited Mar. 23, 2009); *see also* Beth Simone Noveck, *"Peer to Patent": Collective Intelligence, Open Review, and Patent Reform*, 20 HARV. J.L. & TECH. 123, 127–28 (2006) (describing a pilot approach to allow public input to the U.S. Patent and Trademark Office's patent application examination process, where the input from scientists and technologists is primarily via submittal of prior art).

1. Implied License

The ominous threat from patent law arises from activity by Microsoft, but includes the fear of other assertions of patent rights, such as that exemplified by the company in the *Jacobsen* case.[156] Microsoft has asserted that 235 of its patents are infringed by certain high-profile FOSS applications.[157] It has arranged license agreements with some end users in conjunction with their FOSS distributors, which some speculate is based on this assertion.[158] At the other end of the corporate spectrum in size and scope, the company that originally threatened Robert Jacobsen did so with patent rights. Jacobsen's group found itself in the paradigmatic worst-case patent scenario of a FOSS community: a for-profit competitor might have patent leverage over the group. If the scope of the patent claims allowed for it, Jacobsen's group could perhaps redesign and reprogram their FOSS application to not infringe in the future, leaving the interesting question of

---

156. Complaint for Declaratory Judgment, for Violations of Antitrust Laws, California Business and Professions Code § 17200, and Lanham Act, and for Libel, Demand for Jury Trial at 25–27, *Jacobsen I*, 2007 WL 2358628 (N.D. Cal. Aug. 17, 2007) (No. C 06-01905 JSW), *available at* http://www.jmri.org/k/docket/1.pdf (asserting in declaratory judgment motion that Katzer's patent is unenforceable and not infringed by the software Jacobsen's hobbyist group developed as FOSS); Letter from Kevin L. Russell, Attorney for Matthew Katzer, to Robert G. Jacobsen, JMRI Project (Mar. 8, 2005), *available at* http://www.jmri.org/k/correspondence/20050308-KAM.pdf (asserting patent infringement by the FOSS group's software).

157. Babcock, *supra* note 99.

158. Microsoft entered into an arrangement with Novell relating to patents. Microsoft-Novell Patent Cooperation Agreement (Nov. 2, 2006), *available at* http://www.sec.gov/Archives/edgar/data/758004/000095013407012375/f26782exv10w35.htm. Without delving into the details of the patent licensing provisions in the agreement here, what happened next is that the FSF reacted by inserting a provision into GPLv3. The license was still in draft form at the time of the Microsoft-Novell agreement. The provision sought to diminish future possibilities of the Microsoft-Novell arrangement, but grandfathered in the arrangement.

> You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

GPLv3, *supra* note 15, § 11; *see also* Novell, Novell Answers Questions from the Community, http://www.novell.com/linux/microsoft/faq_opensource.html (last visited Mar. 23, 2009) ("Our agreement with Microsoft . . . does not include a patent license or covenant not to sue from Microsoft to Novell . . . . Novell's customers receive a covenant not to sue directly from Microsoft. We have not agreed with Microsoft to any condition that would contradict the conditions of the GPL . . . .").

Some have argued that this GPLv3 provision might suggest an implied license by Microsoft of its patents in conjunction with other aspects of the Microsoft/Novell arrangement. Those assertions and details will also be put aside, and this specialized situation will not be treated below in Part IV.C.2, where a generalized implied scenario is evaluated.

ex post evaluation for patent infringement damages when the FOSS is distributed for free.[159] In the *Jacobsen* case, according to one commentator, the patent threat diminished as the case developed.[160]

The conceivable implied license issues under patent law involving FOSS distribution are numerous. Some cases taxonomize the law of implied license because the case outcomes are highly specific to the facts and circumstances and outcomes are often based on different approaches.[161] The common denominator for the patent holder or licensor is the potential diminishment of whatever appropriation leverage she expected with the patent or license. The diminishment is due to a successful assertion of implied license as a defense. In this essay, the goal is not to exhaustively treat implied license in patent law, but to look at a few scenarios where it might alter appropriability in commercial FOSS.

The first conceivable scenario involves a patent holder that also contributes to a FOSS project when some or all of that software embodies

159. Patent remedies include injunctions and damages. 35 U.S.C. §§ 283–284 (2000). The damages standard starts with the statutory command that damages be "adequate to compensate for the infringement, but in no event less than a reasonable royalty for the use made of the invention by the infringer." *Id.* § 284. Working backward through this language in the context of a FOSS project with overlapping and potentially infringing functionality compared to a patent-enforcing proprietary software product, what are reasonable royalties when the FOSS is distributed without ongoing royalties because the FOSS license disallows that? Reasonable royalties establish a floor to damages, but when it can, a patent suit plaintiff may try to prove lost profits due to lost sales. This raises additional questions about how to count lost sales when FOSS products are in the market and is discussed *infra*, in Part IV.C.2.

160. Lawrence Rosen, Bad Facts Make Good Law: The Jacobsen Case and Open Source 2 (Oct. 1, 2008) (unpublished manuscript), *available at* http://www.rosenlaw.com/Bad FactsMakeGoodLaw.pdf ("It soon turned out that [the company's] patent infringement allegations were bogus.").

161. Most implied license cases involve entangled parties. Thus, one type of implied license is where the patent holder or licensor grants some type of a right for consideration, such as a license statement allowing the licensee to make, sell, and use a described technology (as opposed to merely licensing specific patents). Under the notion that the licensor should not be allowed to derogate from the granted right, if the licensor later acquires a patent from a third party that dominates the technology, the licensee may be found to have an implied license under that after-acquired patent. *See* AMP Inc. v. United States, 389 F.2d 448, 454–55 (Ct. Cl. 1968).

A second type of implied license is akin to finding the intent of the parties to a contract under all the facts and circumstances. A third type is more applicable to nonentangled parties, and thus perhaps especially relevant to FOSS. When a patent holder distributes a technology that has as its only purpose practicing the claimed method or apparatus/composition (either alone or with user or third-party supplied technology), this may raise the implied license defense. Anton/Bauer, Inc. v. PAG, Ltd., 329 F.3d 1343, 1350–53 (Fed. Cir. 2003) (on preliminary injunction posture, manufacturer of battery connecting plate held to have granted an implied license for claims infringed when the manufacturer's female plates are mated with third-party-manufactured male connector plates); Met-Coil Sys. Corp. v. Korners Unlimited, Inc., 803 F.2d 684, 687 (Fed. Cir. 1986) ("A patent owner's unrestricted sales of a machine useful only in performing the claimed process and producing the claimed product plainly indicate that the grant of a license should be inferred." (internal quotation marks omitted)).

the claims.[162] This scenario was often discussed in relation to GPLv2 because that widely used license does not state an explicit patent grant, so commentators suggested the possibility of an implied license in conjunction with the GPL.[163] Whatever FOSS license is involved, its text will typically impact the implied license analysis, as will communications or statements among the patent holder/licensor and the licensees. The inference potentially leading to a finding of implied license in this first scenario is that the patent holder should not be allowed to use patent rights to derogate from the FOSS copyright license granted to the public.[164]

The second scenario is an adjustment of the first, where the patent claims are not embodied by the FOSS project to which the holder contributes/supports. Perhaps the FOSS software is an element of what is claimed, or perhaps not. The point in this scenario is that it might be possible for users of the FOSS to incorporate it into other information technology which then infringes the holder's patent. The other technology might infringe alone, or infringes because of the combination with the FOSS. This fact would matter in an implied license analysis, as would the degree of specificity of the software contributed by the patent holder: the more that it is attuned to a single purpose whereby it needs to be fitted with other software, the more likely a finding of implied license when the combination infringes the patent claims of the contributing holder.[165] On the other hand, if the patent claim scope is nonoverlapping or technologically remote from the FOSS, the inferential logic diminishes for finding an implied license defense without something more, such as conduct or statements by the patent holder.

While MetaCarta is not assumed to necessarily fit in either situation one or two, it represents a third scenario because it expressly repudiated any implied license for any party's patents in the FOSS license it constructed for the three FOSS projects it supports.[166] While such an express

162. When the hypothetical patent holder is the primary developer of a FOSS project that embodies the claims and that has many complements, then this configuration resembles a theoretical model for optimizing FOSS licensing terms for a platform progenitor. *See* Geoffrey Parker & Marshall W. Van Alstyne, Innovation Through Optimal Licensing in Free Markets and Free Software (Sept. 2005) (unpublished manuscript), *available at* http://ssrn.com/abstract=639165 (suggesting a period of exclusive control for the platform originator for improvements to the platform, followed by the remainder of the platform life having an open source approach to improvements).

163. ADAM PUGH & LAURA A. MAJERUS, FENWICK & WEST LLP, POTENTIAL DEFENSES OF IMPLIED PATENT LICENSES UNDER THE GPL (2006), *available at* http://www.fenwick.com/docstore/Publications/IP/potential_defenses.pdf.

164. One question about this inference is whether the right granted from the FOSS licensor to the public as FOSS licensee needs to be supported with consideration, which in its traditional form may be lacking. *See id.* (arguing that the possibility of downstream improvements being redistributed for public availability may substitute for, or act as, consideration to certify the right not to be derogated).

165. *Anton/Bauer*, 329 F.3d at 1350–53.

166. ClearBSD, *supra* note 150 ("NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS LICENSE."). *But see* Nat'l Rubber Mach. Co. v. McNeil Mach. & Eng'g Co., 132 F.2d 436, 438 (6th Cir. 1942) (license

repudiation is likely effective, in an implied license analysis sometimes other facts and circumstances can overwhelm such an express statement. If we assume that MetaCarta's situation is close to scenario one, where its contributed software embodies, or nearly embodies, the patent claims, then the express repudiation is a factor the court might balance against another fact: inbound contributors may grant patent licenses to the project. Why aren't these passed on? The question gives one pause, and might give a court pause. I do not intend to conclude on this issue in the abstract. The real point is the pliability of the implied license analysis when mixing FOSS licenses with contributor conduct and communications.[167] Such pliability allows greater leeway for a court to shape outcomes in light of perceived benefits from different modes of production and distribution, that is, proprietary software versus FOSS, or a hybridized mixture of the two.

To the extent implied patent license case law develops around FOSS situations, the cases may share a characteristic often not found in the traditional implied license cases: nonentangled parties.[168] This results from the public nature of FOSS and its distribution approach that often allows any party to download the software without identification. Nonentanglement cases will not have in evidence any communications or actions between the parties, so the cases will turn on the public statements of the FOSS contributing/supporting patent holder, the terms of the FOSS license, whether the developer community was aware of the patents, whether the patent holder helped shape the license or community, and claim scope in relation to the FOSS. Courts should add to this mix a frank assessment of the public benefit spillovers for the FOSS at issue, which will turn in part on information about its licensing model, functionality, developer community, and user base. Among these inquiries, user base is a subject about which it is often hard to obtain good information for many FOSS projects. This problem is also an issue for the topic of the next subsection: patent law injunctive remedies for commercial FOSS.

2. Injunctive Remedies

The competitor exclusionary power of a patent is perhaps at its height with an injunction, and therefore the threat of an injunction is an important

agreement schedule excluded patent from license grant, but other facts and circumstances allowed for an implied license).

167. FOSS licenses do not have "integration" clauses, or at least, I do not recall having seen an integration clause in a FOSS license, thus, to the extent courts use contract construction principles to interpret the conditions of FOSS licenses, it seems that evidence of statements and conduct associated with the license promulgation and/or the contributed software will be relevant and heard by the court.

168. By the term nonentangled parties, I do not mean nonpracticing entities. These entities, sometimes called "patent trolls," are not in the market, whereas the scenarios discussed in the implied license subsection assume a patent holder engaged in commercial FOSS and therefore in the market.

mechanism of appropriation.[169] In 2006, the potency of the injunction diminished under the U.S. Supreme Court's opinion in *eBay, Inc. v. MercExchange, L.L.C.*[170] The case instructs courts to use a four-factor test to determine whether to grant an injunction to a successful patent infringement plaintiff. FOSS casts a different light on all four factors. A conceivable suit is by a proprietary software plaintiff against either a commercial FOSS defendant or against a community-driven FOSS project. Also conceivable is a suit by a commercial FOSS entity against a community-driven project, but I assume that no community projects will ever hold patents to act as a plaintiff.

To proceed, this subsection arrays the four factors against four potential suits, where proprietary software uses the abbreviation "Prop," commercial FOSS goes by "cFOSS," and community-driven FOSS uses "tFOSS" for traditional FOSS. The purpose of the resulting table is to express some high-level, abbreviated items that a court evaluating each factor should entertain as an additional consideration due to the presence of FOSS in the suit. The premise behind these considerations is to account for the differences of FOSS as a mode of knowledge production and its public benefit spillovers. In other words, the table presents additions to whatever would be considered typically. If there are no additions, the annotation reads "no change."

To properly evaluate several of the factors, FOSS projects need better data about their user base. The appellate court in the *Jacobsen* case mentioned in a footnote the general benefits of FOSS as a mode of knowledge production and distribution. The legal inquiry before the court did not require any specific quantification of that public benefit by looking at the user base of Robert Jacobsen's group. When such an inquiry becomes important because, many, if not most, FOSS projects have only a download count. It is not rational to equate download counts to installed users, and there may be no good way to reliably prove to a court that a user base number derived from download counts is believable. The user base data is often insufficient and this could negatively impact the proof needed for a FOSS project involved in some future patent litigation.

169. Mark A. Lemley & Carl Shapiro, *Patent Holdup and Royalty Stacking*, 85 TEX. L. REV. 1991, 1992–93 (2007).

170. 547 U.S. 388 (2006); *see also* Bernard H. Chao, *After* ebay, Inc. v. MercExchange*: The Changing Landscape for Patent Remedies*, 9 MINN. J. L. SCI. & TECH. 543 (2008).

Table 1: Injunction Factor Considerations for FOSS

P	D	(1) Irreparable Injury to P	(2) Law Damages Inadequate for P	(3) Balance of Hardships	(4) Public Interest Not Disserved by Permanent Injury
Prop	**cFOSS**	No change	No change[171]	Account for scope and reach of cFOSS software, perhaps with less weight on D's commercial customers	Impact on cFOSS developer community and user base, perhaps with less consideration for D's commercial customers
cFOSS	**Prop**	No change	No change, unless triggered by a patent peace clause[172]	No change	No change
Prop	**tFOSS**	No sales and marketing by D; not intended as a commercial sales threat[173]	Users of the tFOSS software are not necessarily sales that would have been made by P	Account for scope and reach of tFOSS software[174]	Impact on D's developer community and user base; impact on FOSS licensing system generally

171. This place in the matrix illustrates an advantage of commercial FOSS: it has the possibility of paying damages for the past, and it may have sufficient financial resources to secure a settlement. Hopefully, commercial FOSS companies would obtain settlements applicable across the board for at least community-driven FOSS along with the settlement for itself. In addition, to the extent some courts may use *eBay, Inc. v. MercExchange, L.L.C.* as a basis to award "future-looking damages" or "compulsory" licenses, a commercial FOSS company may also be able to fund this as well. In contrast, a community-driven FOSS project is unlikely to be able to pay anything without severe financial impact on the core developers and other named defendants from the community. This includes limited ability to fund a defense in the patent infringement suit, much less pay damages.

172. The commercial FOSS entity's impetuses to sue a proprietary software company could arise as the result of a retaliation need in light of a patent peace clause in a FOSS license. Assume that the proprietary software company was a user of the commercial FOSS entity's community-supported FOSS and it sued another user on a patent. Under the patent peace clause, the suit might terminate the proprietary company's rights under the FOSS license. To protect the users, the commercial FOSS entity may need to initiate a patent suit against the now-unlicensed proprietary software company. Note that the extent to which this retaliation suit gains leverage over the proprietary software company depends on the degree of importance of the FOSS for its operations.

173. FOSS users are merely theoretical proprietary software customers because the proprietary software provider has perhaps been unwilling or unable to price-discriminate to capture this group. In evaluating the irreparable harm factor generally, the nature of the

P	D	(1) Irreparable Injury to P	(2) Law Damages Inadequate for P	(3) Balance of Hardships	(4) Public Interest Not Disserved by Permanent Injury
cFOSS	**tFOSS**	Evaluate comparative diminishment between the two different FOSS communities	Account for P's voluntary engagement with FOSS, signaling acceptance of unusual appropriability mixes	Account for scope and reach of tFOSS software; account for P's voluntary engagement with FOSS	Assuming two different FOSS projects, impact on D's developer community and user base; impact on FOSS licensing system generally

Here is an example showing how insufficient user information could be a problem. Assume that the patent plaintiff is a proprietary software company suing a community-driven FOSS project, where the software has identical functionality. Further assume that the company's sales dropped from 30,000 licenses/year in prior years to 15,000 licenses/year in the last year for internal causes, but it does not realize the cause. During this prior year the community-driven FOSS project posted its first compete version and it had 10,000 downloads, of which it can verify that 8000 were from computers in the United States, but it can verify no more—it does not know what percentage of those 8000 U.S. downloads are active users. Assuming a successful patent infringement action, plaintiff's damages model will

competition in a market is relevant, particularly as to the number of competitors. TruePosition Inc. v. Andrew Corp., 568 F. Supp. 2d 500, 531–32 (D. Del. 2008) ("Courts awarding permanent injunctions typically do so under circumstances where plaintiff practices its invention and is a direct market competitor. Plaintiffs are also frequently successful when their patented technology is at the core of its business, and/or where the market for the patented technology is volatile or still developing." (footnotes omitted)). Whether a traditional (tFOSS) community should ever be modeled as a competitor in the *eBay* analysis is questionable given the public benefit spillover potential of tFOSS communities.

174. Accounting for the reach of the user base for tFOSS software is consistent with another area of law that sometimes bears on user rights and license agreements for intellectual property: § 365(n) of the Bankruptcy Code. 11 U.S.C. § 365(n) (2006). By allowing, under certain conditions, an option for the licensee to elect to continue to use the intellectual property, the code gives recognition to the reliance interest within the group of licensee users. Robert T. Canavan, *Unsolved Mysteries of Section 365(N)—When a Bankrupt Technology Licensor Rejects an Agreement Granting Rights to Future Improvements*, 21 SETON HALL L. REV. 800, 812–13 (2000) (noting that a nondebtor licensee may elect to continue its licensee status, thereby protecting its investment in manufacturing or other capacity dependent on the license). My thanks to Sharon Sandeen for this point. The analysis in the table contemplates the tFOSS developers as patent defendants rather than noncontributing users. However, the nondeveloping users are also potential patent infringement defendants. In either case, an injunction diminishes the productive capacity of both developer-users and nondeveloper-users of the tFOSS software to whatever extent their activity has become dependent on the software.

likely propose that at least 8000 licenses are its "lost sales" attributable to the FOSS community's infringement. In reality, there may be only 800, or 80, active users. Only the true active users seem like a plausible inclusion into the lost sales damages model. When the number is low, the FOSS projects wants the true number because it reduces the damages amount potentially payable.

This example can also apply to injunctive relief. The recommendation in Table 1 for the public benefit factor is to assess the impact on the FOSS user base. How can this be done before a court if a reliable count of active users is not available? Generally, for a community-driven project, more users means more public benefit resulting from the FOSS. If the FOSS project could show that 5000 of the 8000 U.S. downloads are active users who have come to rely on the software, and royalty-free use of it, it should give a court pause before implementing a permanent injunction.

The two scenarios of this example suggest that FOSS development practices begin to explore technological techniques to gain some minimal information about their active user base.[175] If privacy concerns are an issue, the information can be nonidentifying of the users so long as it is collected with a technique provable in court as reliable. Collecting this information is not necessarily important only for GPL-style FOSS that remains in community development. It could be important for attribution-only FOSS that is embedded in proprietary software. Thus, such data could be particularly helpful to argue the public benefit prong of the four-factor permanent injunction test if a FOSS community could show that among the hypothetical 8000 U.S. downloads, 1000 were embedded instances of the software in a proprietary product.[176]

Many in the FOSS movement might argue that the last row in Table 1 above is inconceivable given the political backlash a commercial FOSS entity would suffer if it sued a community-driven FOSS project. This argument might extend to suggest that the possibility of such a suit is a reason why patent-based mechanisms of appropriability are a poor choice for the appropriation mix of a commercial FOSS firm. This essay's premise is to acknowledge these views but assume that patent issues such as implied license questions or injunctions are going to arise eventually, even if they are most likely to arise in one of the scenarios in the top three rows that involve a proprietary software company. If the prediction is correct that such issues will arise, the important fallback step will be to have courts recognize the successes and aspirations of the FOSS movement in order to

175. *See* VON HIPPEL, *supra* note 14, at 88 ("It can be difficult to track what visitors to an information commons take away and reuse, and there is as yet very little empirical information on this important matter.").

176. The recommendation for some type of technical apparatus to collect this information should not overlook the possible need to account for the collection activity in the FOSS licenses. For instance, in the embedded FOSS example, the attribution-only license would probably need to add a condition that the user not remove a particular software object, or file of code, that implements the collection of active user base information.

understand its narrative for knowledge production and public benefit in the context of the case before it.

CONCLUSION

In the puzzle of appropriability for new knowledge outputs from information technology, FOSS is a box of paradoxes. The oft-noted paradox is that FOSS licensing uses intellectual property rights, most prominently copyright, to imbrue software with conditions defeating not only trade secrecy in the code but also the conventional use-restricting deployment of copyright itself. Some FOSS licenses extend this dynamic to patent rights for those involved with the community underlying the software. FOSS licensing thus enhances public accessibility for the knowledge contained in its software source code. In its strongest form, under copyleft licenses, public accessibility may be cemented for the future of the code. Noncopyleft licenses such as attribution-only licenses often allow direct commercialization of the code under a proprietary software model. Thus, paradoxically, a robust FOSS ecology may subsidize both proprietary vendors and new business models such as advertising-supported software or Internet-delivered software as a service.

Another paradox relevant to this essay is that while FOSS licensing has spawned numerous benefits within information technology, including direct knowledge generation of new collaborative techniques, as well as other important spillovers, it may not be doing as well in producing innovative software in the sense of new nonplatform functionality. To the extent this is true, it may be path dependent because proprietary software came first. To the extent courts deal with future licensing issues related to FOSS, their perspective on this issue is one of several considerations in the narrative about FOSS that may influence outcomes or the policy that informs outcomes.

The FOSS narrative has traditionally been one of volunteer-centric projects. That narrative dominated in the recent important FOSS case of *Jacobsen v. Katzer*. With increasing hybridization, however, of FOSS with appropriation methods akin to the world of proprietary software, the narrative told to courts may change. The appearance of patents for competitor exclusion appearing among the appropriation methods may be particularly noticeable and narrative-influencing. This observation rests in a background of two realities: great acrimony by the FOSS movement with respect to patent protection for software; and the possibility that proprietary software companies will use patents against FOSS communities. The latter possibility juxtaposes two very different knowledge production paradigms when FOSS is represented by its traditional narrative. But when commercial FOSS of a hybridized nature is involved, courts will need to closely evaluate the FOSS elements in order to account for their contribution to the public benefit within the context of the licensing dispute at issue. In the patent licensing context, this will require a nuanced view of issues such as injunctions that may curtail or chill FOSS development and

issues related to implied license doctrines under patent licensing. Even while deplorable by much of the FOSS movement, it seems inevitable that an increase in commercial FOSS will result in an increased entanglement with patent law. This cries out for a careful accounting of the beneficial influences from FOSS to information technology within the context of each case, because increasing deployment of commercial FOSS also seems inevitable.