

## Software Patents and Open Source: *The Battle Over Intellectual Property Rights*

DAVID S. EVANS<sup>†</sup> & ANNE LAYNE-FARRAR<sup>‡</sup>

### ABSTRACT

In the wake of a series of court cases extending patents to software, open-source software proponents have proposed a number of arguments for limiting or even eliminating software patents. In particular, they claim that the U.S. Patent and Trademark Office (USPTO) has done a poor job of reviewing software patent applications, resulting in obvious, trivial patents. Open-source proponents also maintain that software patents hinder the standards-setting process important for high-technology industries, and that patents will lead to intellectual property rights “thickets” that slow down or stop the innovative process in the software industry. We evaluate these claims, examining relevant empirical evidence where available. While it is clear that problems exist with the patent-granting process, they do not rise to the level of justifying a ban on software patents. Instead, other reasonable—and far less drastic—measures are available. The USPTO has already begun reforms that should improve its software patent-review process. As for patent thickets, theory suggests they could form in the software industry, but empirical evidence suggests that in fact this has not occurred. Moreover, tools such as patent pools and cross-licensing can increase innovation sharing and are available to limit the development of thickets. While the academic literature is still debating the link between patents and innovation, patents have been shown to have some positive effects, including increased venture capital funding for small firms. In the end, reform is far more attractive than abolition, because it retains the good while minimizing the bad.

---

© 2004 Virginia Journal of Law & Technology Association, at <http://www.vjolt.net>. Use paragraph numbers for pinpoint citations.

<sup>†</sup> Evans is with LECG, LLC in London, England and Cambridge, Massachusetts.

<sup>‡</sup> Layne-Farrar is with LECG, LLC in Chicago, Illinois. The authors would like to thank Bernard Reddy and Sean MacLeod for helpful comments and suggestions. They are also grateful to Microsoft for supporting the research upon which this paper is based.

## TABLE OF CONTENTS

I.	Introduction.....	2
II.	A Survey of the Software Landscape.....	3
	A. An Open-Source Primer.....	3
	B. The Evolution of Software Patents .....	5
	1. Software Patents in the United States .....	5
	2. Software Patents in Europe .....	7
	C. Software Licenses: Expressions of the Patent Conflict .....	8
III.	Open-Source Arguments against Patents.....	11
	A. Flaws in the Patent Process.....	11
	1. The U.S. Patent and Trademark Office Issues Obvious, Non-Original, Trivial Software Patents.....	11
	a. Anecdotal Evidence Can Be Misleading.....	12
	b. The Process Is Improving.....	14
	c. Empirical Evidence Suggests Some Criticisms May Be Unfounded.....	15
	2. Patents can hold standards hostage .....	15
	a. Standard setting organizations can help .....	17
	b. USPTO reforms should lessen the problem .....	18
	B. Economic Theory and the Justification of Patents.....	19
	1. Software was innovative before patenting .....	19
	a. The industry has changed .....	20
	2. Software development is a sequential and cumulative process making patent thickets likely .....	21
	a. Patent thickets are not inevitable.....	23
	b. There are other ways to prevent patent thickets.....	24
	3. Patents can reduce the software “commons” .....	25
	a. Future innovation must be considered when measuring the software “commons”.....	25
	b. The alternative to patents is worse .....	26
IV.	Conclusion .....	27



### I. INTRODUCTION

¶1 Hard-line open-source software proponents have been waging war against intellectual property rights since the dawn of the movement in the 1980s.<sup>1</sup> Early skirmishes involved trade secret law and the right to keep software users from seeing the

---

1. Some members of this community object to the term “intellectual property,” preferring instead that all discussions of the subject separately consider patent, copyright, trademark, and trade secrets. Richard Stallman, *The GNU GPL and the American Way*, available at <http://www.gnu.org/philosophy/gpl-americanway.html> (last updated Feb. 27, 2004).

source code, or instructions for a program.<sup>2</sup> Later battles involved copyright, with open source crafting its own copyright licensing scheme dubbed “copyleft.”<sup>3</sup> The newest front of the war being waged by the hard-line open-source software proponents is over software patents.

¶2 This article examines the arguments made by open-source software proponents concerning limiting or eliminating software patents. To put today’s debate in context, Section II provides some background. Part A briefly explains open-source software and describes its coexistence with traditional proprietary software.<sup>4</sup> Part B then summarizes the pivotal court cases that enabled developers to obtain patents on software. Originally, software was viewed as a collection of pure mathematical algorithms and, as such, not eligible for patent protection. That view changed incrementally from the early 1970s through the 1990s; however, it was not until 1994 that patents on stand-alone software could be granted. Thus, the open-source fight against software patents has only recently begun. Closing Section II, Part C describes a few prominent open-source licenses and how they deal with—or try to prevent—patented software. Section III explains and discusses the arguments made by open-source advocates concerning software patenting. While many of these arguments hold some kernel of truth, they do not rise to the level of justifying a ban on software patents. Other more sensible alternatives are available for fixing whatever problems exist in the patenting process. Section IV concludes.

## II. A SURVEY OF THE SOFTWARE LANDSCAPE

### A. An Open-Source Primer

¶3 Open source is, in essence, a way of creating and distributing software.<sup>5</sup> Rather than keep the human-readable program instructions (called source code) hidden from users, as traditional software companies such as Intuit and Apple typically do, open-source programs make the source code available to one and all. This access enables users

---

2. See, e.g., Richard M. Stallman, Free Software: Freedom and Cooperation, Speech delivered at New York University (May 29, 2001), available at <http://www.fsf.org/events/rms-nyu-2001-transcript.txt>.

3. As the Free Software Foundation describes it, “copyleft (very simply stated) is the rule that when redistributing the program, you cannot add restrictions to deny other people the central freedoms” of running the program for any purpose, studying how it works and adapting it to their needs, redistributing copies, or improving the program and distributing the revised version. Free Software Foundation, *The Free Software Definition*, available at <http://www.fsf.org/philosophy/free-sw.html> (last updated Feb. 19, 2004). For a more precise definition of copyleft, see Free Software Foundation, *What is Copyleft?*, available at <http://www.fsf.org/copyleft/copyleft.html> (last updated Mar. 2, 2004). Copyleft, of course, cannot be enforced without the rights granted by copyright law. See David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241, 287.

4. We refer to software generally sold by for-profit software developers as proprietary. Proprietary software is what most of us are familiar with, including Microsoft Word for Windows or Intuit’s QuickBooks. We use the term “open source” to describe software that is made readily available in the form of source code.

5. For a general discussion of open-source software, see DAVID S. EVANS, OPEN-SOURCE SOFTWARE POSES CHALLENGES FOR LEGAL AND PUBLIC POLICY, (Wash. Legal Found., Legal Backgrounder, Jan. 31, 2003).

skilled in programming to become de facto software developers by adding to or modifying the software code and then redistributing it. Open-source software is not licensed to individual users or companies in the conventional fashion—users can share the software with others if they choose. As we shall see in the next section, however, open-source software is not rule-free since it imposes certain restrictions on users.

¶4 Despite their oil and water properties, open-source and proprietary software have coexisted for over two decades—and each has something to offer software users.<sup>6</sup> In the early days of computing, software and hardware were often combined in a single package. IBM, for example, sold its mainframe computers with hardware-specific operating system software. A separate software industry did not emerge until the early 1970s, and did not really burgeon until the advent of the PC in the late 1970s. Not long thereafter, the open-source movement as we now know it emerged.<sup>7</sup>

¶5 In 1984, Richard Stallman founded the Free Software Foundation (FSF).<sup>8</sup> The inspiration occurred in 1980, when Stallman was refused access to the source code for software used to access a printer at the MIT Artificial Intelligence Lab.<sup>9</sup> His indignation over the incident eventually led to his morals-based view of software sharing and the founding of the FSF. As Stallman expresses it, “[m]y work on free software is motivated by an idealistic goal: spreading freedom and cooperation. I want to encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better.”<sup>10</sup>

¶6 One of the best-known open-source programs is Linux, an operating system developed by Linus Torvalds beginning in 1991. Torvalds patterned Linux after Unix, an operating system developed by AT&T in 1969 which is still used extensively, particularly on server computers.<sup>11</sup> Today, Linux is a highly popular and well-known

6. For a comparison of the two forms of software, see David S. Evans & Bernard J. Reddy, *Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem*, 9 MICH. TELECOMM. & TECH. L. REV. 313, 356-65 (2003).

7. In the early 1970s, programmers in academic settings shared programs and swapped sections of source code. This could be viewed as a precursor to today’s open-source movement. See Stallman, *supra* note 2. Stallman ardently rejects the label “open source,” preferring instead “free software.” *Id.* While the semantics of the movement tend to hold a great deal of meaning for movement participants, for our purposes there is little distinction between the two monikers. Therefore, we refer to all software that is distributed with its source code (and with users allowed to redistribute that source code) as open source.

8. The Free Software Foundation did not formally exist at the beginning, but it is essentially an outgrowth of the GNU Project, which was founded in 1984. See Free Software Foundation, *GNU’s Not Unix!*, available at <http://www.gnu.org/home.html> (last updated Mar. 22, 2004).

9. Stallman, *supra* note 2.

10. See Richard Stallman, *Copyleft: Pragmatic Idealism*, Free Software Foundation, available at <http://www.fsf.org/philosophy/pragmatic.html> (last updated Feb. 21, 2004).

11. What is commonly known as the “Linux” operating system is really much more than just the “Linux” kernel (or core set of instructions), which is what Torvalds developed. The Free Software Foundation had been developing a free Unix-like system, called GNU, and was essentially just lacking a kernel (arguably the most important part) at the time Linux was completed. In a contemporary Linux distribution, the amount of code from the GNU Project probably exceeds that in the Linux kernel (although much of the code comes from many other sources), and the Free Software Foundation takes great pains to encourage use of the name “GNU/Linux.” See Richard Stallman, *Linux and the GNU Project*, Free

open-source program—there was even an advertisement for it featuring boxing champ Muhammad Ali during the 2004 Super Bowl (courtesy of IBM). As such, Linux has done much to raise the profile of open source. In fact, Linux is largely responsible for exposing Wall Street and mainstream industry to open-source software. Prior to Linux, the most widely used open-source programs were back-end programs targeted primarily at developers and website administrators.<sup>12</sup> Linux, however, as an operating system for both client computers (PCs) and server computers, is challenging proprietary software closer to its home turf. As a result, the battle lines between proprietary and open-source software sharpened in the late 1990s.

## B. The Evolution of Software Patents

¶ 7 Another factor that heightened the tensions between open-source and proprietary software in the 1990s was the evolution of patent law. Software patents are a relatively recent phenomenon in this country and are even newer in Europe.

### 1. Software Patents in the United States

¶ 8 Through the 1970s, software was considered equivalent to mathematical algorithms or laws of nature, and thus was not patentable.<sup>13</sup> The Supreme Court upheld this interpretation in 1972 when it struck down a patent on a new and faster process for converting decimal numbers to binary numbers.<sup>14</sup> A handful of patents granted later in the 1970s could be classified as software patents today, but by and large the ruling stood unchallenged for the next decade.

¶ 9 A Supreme Court ruling in 1981 drastically changed software patenting.<sup>15</sup> While it did not fully establish the software patenting standards in place today, the *Diamond v. Diehr* decision set the ball rolling for later precedent-setting decisions that further extended patents to software. Diehr had developed a process for “molding raw, uncured synthetic rubber into cured precision products,” which involved the use of software in measuring and monitoring the process.<sup>16</sup> In finding the patent valid, the Court’s fundamental reasoning was that “a claim drawn to subject matter otherwise statutory does

---

Software Foundation, available at <http://www.fsf.org/gnu/linux-and-gnu.html> (last updated Feb. 21, 2004). In keeping with normal usage, however, we will use the term “Linux” to mean the operating system, not just the kernel.

12. Apache and Sendmail are two very successful open-source programs, in the sense that they are the most often used programs in their categories. However, few non-IT people have heard of them. See Andrew Leonard, *Apache’s Free Software Warriors*, SALON MAG. (Nov. 1997), available at [http://archive.salon.com/21st/feature/1997/11/cov\\_20feature.html](http://archive.salon.com/21st/feature/1997/11/cov_20feature.html); Andrew Leonard, *You’ve Got Sendmail*, SALON MAG. (Nov. 1997), available at [http://archive.salon.com/21st/feature/1998/12/cov\\_11feature.html](http://archive.salon.com/21st/feature/1998/12/cov_11feature.html).

13. The Patent Act of 1952 defines the following as patentable inventions: “[A]ny new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof.” 35 U.S.C. § 101 (2004). Courts have long held that mathematical formulae, algorithms, and laws of nature are not patentable subject matter under the Patent Act. See *Diamond v. Diehr*, 450 U.S. 175, 186 (1981).

14. *Gottschalk v. Benson*, 409 U.S. 63, 71-72 (1972).

15. *Diamond v. Diehr*, 450 U.S. 175, 185 (1981).

16. *Id.* at 175.

not become non-statutory because a computer is involved.”<sup>17</sup> According to Justice Rehnquist, who wrote for the majority, “[the] claims must be considered as a whole.”<sup>18</sup> The ruling established that the algorithm contained in the patentee’s software was not protected as an abstract idea, but rather in its application in the rubber curing process.<sup>19</sup> As a result, software in isolation remained unpatentable. Nonetheless, the new standard enabled patent drafters to bundle software innovations within patents for broader patenting of larger, more specific processes.

¶10 Another decade passed before the patentability precedent was ratcheted up another notch. In 1994, the Court of Appeals for the Federal Circuit (CAFC) handed down the decision *In re Alappat*, which reversed the U.S. Patent and Trademark Office (USPTO) Board’s invalidation of a patent on “a means for creating a smooth waveform display in a digital oscilloscope.”<sup>20</sup> The Board had acted partly on the grounds that even though the invention was for use in an oscilloscope, the doctrine of equivalents<sup>21</sup> would also protect the use of the same methods on a “general purpose digital computer.”<sup>22</sup> The CAFC, building on *Diehr*, rejected the Board’s reasoning, finding instead that “such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.”<sup>23</sup> This ruling cemented the statutory standing of software patents.

¶11 In 1998, the CAFC made another landmark ruling, this time striking down the long-standing business method exception.<sup>24</sup> In *State Street v. Signature Financial*, the CAFC reversed a district court ruling that had held invalid Signature’s patent on a “data processing system” for use in its business of administering mutual funds.<sup>25</sup> The district court had argued that Signature’s patent was not valid based on the unpatentability of “abstract ideas” and “business methods.”<sup>26</sup> Pulling directly from *Diamond* and *Alappat*, the CAFC rejected the non-statutory claim based on “unpatentable” abstract ideas,<sup>27</sup> but also went further in holding the “business method exception” itself to be invalid.<sup>28</sup> *State Street* opened the door for the recent surge in business method patents and strengthened

---

17. *Id.* at 181.

18. *Id.* at 188. The decision was by narrow majority, 5-4. *Id.* at 176.

19. *Id.* at 191-93.

20. *In re Alappat*, 33 F.3d 1526, 1537 (Fed. Cir. 1994).

21. The doctrine of equivalents is the standard for judging the extent of the protection under § 112 of the patent code, which states: “such claim[s] shall be considered to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.” 35 U.S.C. § 112 (2004).

22. *Alappat*, 33 F.3d at 1545.

23. *Id.*

24. The business method exception was a judicially-created standard for patentability which prevented methods of doing business from gaining patent protection. *See Hotel Sec. Checking Co. v. Lorraine Co.*, 160 F. 467, 469 (2d Cir. 1908).

25. *State St. Bank v. Signature Fin. Group*, 149 F.3d 1368, 1370 (Fed. Cir. 1998).

26. *Id.* at 1373, 1375.

27. *Id.* at 1374.

28. *Id.* at 1375.

the case law precedent protecting patents on software.<sup>29</sup>

## 2. Software Patents in Europe

¶ 12 Europe is currently grappling with whether the EU patent office should recognize software patents in the same way as the U.S. The European debate has been quite strident on both sides of the issue. Many have voiced strong opposition, pointing to problems in the American system as a warning.<sup>30</sup> Open-source proponents, in particular, view the current European policy evaluation as an opportunity to curtail the spread of software patents.<sup>31</sup> However, members of the proprietary software industry worry that anything short of the U.S. standard will leave them at a competitive disadvantage.<sup>32</sup>

¶ 13 Glossing over the intricate details, the EU is considering harmonizing the treatment of computer-implemented inventions.<sup>33</sup> The European and national-level patent systems (which are harmonized on paper) specifically state that computer-implemented inventions are not patentable.<sup>34</sup> Nonetheless, different member states have interpreted these laws in various ways, resulting in, by some estimates, tens of thousands of already granted European software patents.<sup>35</sup> These estimates indicate the European battle, like the recent American one, will not center on whether to patent software, but rather on how to patent it.<sup>36</sup>

¶ 14 While there have been several legal developments involving European patent systems in recent years, the debate appears no closer to a resolution now than at its start

---

29. See JOSH LERNER, WHERE DOES STATE STREET LEAD? A FIRST LOOK AT FINANCIAL PATENTS 1971-2000, at 29 (Nat'l Bureau of Econ. Research, Working Paper No. 7918, 2000).

30. See, e.g., Tom Yager, *Software Patents Set Sail; Of all the American Ideas to Import, Europe Picks This?*, INFOWORLD DAILY NEWS, Oct. 6, 2003, available at 2003 WL 9969290.

31. See, e.g., Research In Europe, *An Open Letter to the European Parliament Concerning the Proposed Directive on the Patentability of Computer-Implemented Inventions* (Aug. 25, 2003), available at <http://www.researchineurope.org/policy/patentdirltr.htm> (a letter from several European economists regarding their concerns about loosening the standard of patentability for software).

32. Andy Reinhardt, *Inventing a Better Patent Law: Can Europe Spur Software Innovation While Safeguarding Intellectual Property?*, BUS. WK., Dec. 22, 2003, at IM5.

33. *Id.*

34. Convention on the Grant of European Patents (European Patent Convention), Oct. 5, 1973, arts. 52(2)(c), 52(3), 13 I.L.M. 268, 285, available at [http://www3.european-patent-office.org/dwld/epc/epc\\_2002\\_v1\\_bm.pdf](http://www3.european-patent-office.org/dwld/epc/epc_2002_v1_bm.pdf).

35. William Echikson & Matthew Newman, *EU Bows to Business, Postpones Software Patent Vote*, DOW JONES INT'L NEWS SERV., Nov. 26, 2003 available at WL 11/26/03 Dow Jones Int'l News 19:31:00.

36. The European Patent Office and national patent offices in the EU member states argue that the European Patent Convention does not ban patents on programs entirely. For instance, algorithms cannot be patented, but applications of an algorithm to a technical problem can. However, the lack of clarity in the text of the Convention has caused confusion regarding what constitutes a valid patent on a computer-implemented invention, leading to divergent case law among the Boards of Appeal of the European Patent Office and the courts of the member states. For example, the UK has categorically rejected patents on methods of doing business, even when the invention offers a clear technical contribution. See John M. Conley, *The International Law of Business Method Patents*, 88 ECON. REV. (Fed. Res. Bank of Atlanta) 15, 27 (2003). In contrast, the European Patent Office allows patents on business methods if the inventor can show a technical contribution to the state of the art. See *id.* at 23-27.

four years ago. The European Commission submitted a draft bill in June 2003 that would have rationalized the divergent national approaches to software patents and would have reassured firms that their current software patents would remain valid.<sup>37</sup> The European Parliament then amended the Commission's version of the bill in September 2003. However, the amendments completely reverse the meaning of the draft bill submitted by the Commission and categorically reject software patents except under very limited circumstances.<sup>38</sup> Moreover, Parliament's amended bill would have nullified nearly all software patents previously issued by the EU.<sup>39</sup> The bill has since moved on to the next stage in the European legal system for further modification. Meanwhile, the Commission has threatened to kill the legislation if the final form resembles the version approved by the Parliament. Instead, the Commission would negotiate directly with national governments about harmonizing software patent policy.<sup>40</sup> The final outcome of this battle will probably not be decided for a long time.

### C. Software Licenses: Expressions of the Patent Conflict

¶ 15 With this brief history of software patent evolution in mind, the next step in understanding how patents clash with open-source software is understanding how software is distributed to end users. Software is not sold, rather it is licensed. The copyright holders retain "ownership" of their software, but they grant to licensees the right to use the software subject to certain restraints. It is the type of restraint placed on users by the corresponding software license that distinguishes proprietary software from open-source software.

¶ 16 One of the most frequently used open-source licenses was drafted by the FSF as a means of promoting open source at the expense of proprietary software: the GNU General Public License, or GPL.<sup>41</sup> In particular, if a program is distributed under the GPL, all source code must be made available, essentially for free. The GPL also stipulates that any user can modify and distribute the program, either in original or modified form. However, any redistribution (whether of the original or modified program) must also be done under the GPL. This condition has been called "viral," because it typically means that once code is licensed under the GPL, any other code that comes in contact with it falls under the GPL as well. Accordingly, the actual text of the license states: "You must cause any work that you distribute or publish, that in whole or

---

37. Committee on Legal Affairs and the Internal Market, European Parliament, *Report on the proposal for a directive of the European Parliament and of the Council on the patentability of computer-related inventions*, (June 18, 2003), available at <http://www2.europarl.eu.int/omk/sipade2?PUBREF=-//EP//NONSGML+REPORT+A5-2003-0238+0+DOC+PDF+V0//EN&L=EN&LEVEL=3&NAV=S&LSTDOC=Y>.

38. See Matthew Broersma, *Patents Directive Wins European Parliament OK*, ZDNET UK, Sept. 24, 2003, available at <http://news.zdnet.co.uk/business/legal/0,39020651,39116642,00.htm>.

39. Matthew Broersma, *Software Patent Limits "Go too Far"*, ZDNET UK, Sept. 26, 2003, available at <http://news.zdnet.co.uk/business/legal/0,39020651,39116709,00.htm>.

40. *Id.*; see Broersma, *supra* note 38.

41. For information on the Free Software Foundation and the GPL, see <http://www.fsf.org> (last visited Apr. 9, 2004).

in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.”<sup>42</sup>

¶ 17 Interpreting just how far the GPL extends itself is debatable.<sup>43</sup> What seems clear is that it is possible for proprietary software to freely interact with a GPL program without being “infected.” Somewhat less clear, but certainly unchallenged, is that application programs, such as word processing software, that run on top of a GPL operating system (such as Linux) do not have to be licensed under the GPL. How much GPL code has to be included in a new program before triggering the viral license is also unclear—is one line of a GPL program enough? The extent of the gray area remains subject to speculation for now because thus far there has been no court ruling on the matter.<sup>44</sup>

¶ 18 Not surprisingly, given its FSF origins, the GPL provisions are intentionally aimed at preventing open-source code from being incorporated into proprietary code. One result of the source code distribution requirements is that programmers are not allowed to charge more for programs than the cost of reproduction, which is typically quite small. If someone charged substantially more than the reproduction cost for GPL software, anyone acquiring the software could then promptly redistribute it, thereby driving the price down to the reproduction cost. The only feasible extra charges are for additional services, such as support or training, or for complementary proprietary programs. For-profit firms that specialize in open-source software distributions attempt to distinguish themselves via the skill of their employees and the level of service offered by their support staffs. For more complicated open-source products that are really packages of open-source programs, such as a complete Linux operating system distribution (which would include Linux and ancillary software), for-profit firms can differentiate themselves by the specific combination of included add-on utilities, the quality of the installation process including instruction manuals, and subsequent support such as on-line help facilities.

¶ 19 The GPL explicitly singles out patents. If patented code is used in a GPL program, the patent must be automatically licensed for everyone’s free use. In particular,

---

42. Free Software Foundation, *GNU General Public License § 2(b)* (Version 2, 1991), available at <http://www.gnu.org/copyleft/gpl.html> (last updated Mar. 5, 2004).

43. See, e.g., Patrick K. Bobko, *Linux and General Public Licenses: Can Copyright Keep “Open Source” Software Free?*, 28 *AIPLA Q.J.* 81, 96-97 (2000); Shawn W. Potter, *Opening Up to Open Source*, 6 *RICH. J.L. & TECH.* 24, ¶¶ 61-5 (2000), at <http://law.richmond.edu/jolt/v6i5/article3.html>; Jeff C. Dodd & Brian Martin, *Building A Cathedral Over the Bazaar: A Preliminary View of Certain Licensing Practices in the Open Source and Free Software Communities* 13, 43 (unpublished working paper) (on file with journal); Krysten Crawford, *General Public Larceny?*, *CORP. COUNS.*, Dec. 2003, at 81.

44. In a lawsuit against IBM, SCO Group Inc. challenged the constitutionality and enforceability of the GPL but later dropped the charges. See Jim Kerstetter, *The Most Hated Company in Tech: SCO’s Huge Linux Suit Against IBM is a Long Shot that may Yield Nothing but Bile*, *BUS. WK.*, Feb. 2, 2004, at 78. The remainder of the case is still pending. While SCO’s stock has risen by many multiples, the prevailing industry (although not necessarily legal) opinion is that the case has little merit. *Id.* See also *SCO’s McBride Backs Down On “GPL is Unconstitutional” Claim*, *LINUXWORLD.COM*, Apr. 29, 2004, available at <http://www.linuxworld.com/story/44643.htm>.

§ 7 of the GPL states:

If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.<sup>45</sup>

¶ 20 This is of course not an issue when dealing with one's own patents since the patent owner can license their use in connection with the GPL software. However, problems arise when a patent licensee combines the licensed code with GPL code, thereby releasing someone else's patented software to the public. Compulsory royalty-free licensing essentially negates a patent—no one can be excluded and no profit can be earned to recover research or development costs. This, of course, is precisely the goal of the Free Software Foundation.

¶ 21 Linux distributor Red Hat recently applied for two software patents, receiving mixed reactions in the open-source community.<sup>46</sup> To assuage the community's concerns, Red Hat publicly declared GPL-only licenses for the patents.<sup>47</sup> In effect, these patents are automatically licensed for GPL software, while requiring developers issuing software under a non-viral license (such as proprietary and even certain open-source licenses) to negotiate with the patent holder. Special arrangements like this, however, do not alleviate the tension between open-source software and traditional patents.

¶ 22 In contrast to the anti-proprietary, anti-business stance of the GPL, other open-source licenses offer greater commercial flexibility. For example, the Berkeley Software Distribution (BSD) License allows users to redistribute code in a binary form that only computers, not programmers, can understand.<sup>48</sup> Moreover, the BSD License and the other open-source licenses patterned after it permit patented technology to be used with open-source software. In the typical patent case, the holder (or licensee) of a patented technology would use that technology in conjunction with BSD code and distribute only the binary code for the program, just as if the entire program were proprietary.<sup>49</sup>

---

45. See *GNU General Public License*, *supra* note 42, § 7.

46. Matthew Broersma, *Red Hat Defends Controversial Patent Applications*, ZDNET UK, May 31, 2002, available at <http://news.zdnet.co.uk/business/0,39020645,2111257,00.htm>.

47. Red Hat, Inc., *Statement of Position and our Promise on Software Patents* (2004), available at [http://www.redhat.com/legal/patent\\_policy.html](http://www.redhat.com/legal/patent_policy.html).

48. The Regents of the University of California, *The 4.BSD Copyright* (1994), available at <http://www.freebsd.org/copyright/license.html> (last modified Aug. 31, 2003).

49. Parts of Apple's OS X Server are based on BSD Unix, and while Apple released portions of the source code under open-source licenses, it was able to withhold the code for its user interface. See Stephen Shankland, *Apple Opens Parts of its OS*, CNET NEWS.COM, Mar. 16, 1999, available at [http://news.com.com/2100-1001\\_3-223086.html](http://news.com.com/2100-1001_3-223086.html).

### III. OPEN-SOURCE ARGUMENTS AGAINST PATENTS

¶23 The free software proponents (exemplified by Stallman) argue that all proprietary software is bad. Patents are particularly insidious because they prevent open-source software programmers from doing what they want. Other open-source proponents, with less explicit ideological beliefs, make what some might consider more practical arguments against software patents, including those discussed in this section.

#### A. Flaws in the Patent Process

¶24 One category of complaint can best be described as procedural. It argues the following line of reasoning: software patents are often not awarded properly and can be misused, thus they should be eliminated altogether. We explore two criticisms of that nature. The first is that the USPTO is ill-equipped to grant software patents, thus it should not grant them at all. The second is that patents can interfere with the process of standard setting—crucial to software and the computer industry in general—and so should not be granted for software.

##### 1. The U.S. Patent and Trademark Office Issues Obvious, Non-Original, Trivial Software Patents

¶25 Considerable discussion regarding software patents, especially in the popular press, centers on what are claimed to be particularly obvious, trivial software patents.<sup>50</sup> Critics note that, unlike most other scientific and engineering fields, many innovations in the software industry are not published in industry or academic journals.<sup>51</sup> Patent applications require listing of relevant prior art—existing inventions that the new innovation draws on<sup>52</sup>—but if a significant portion of the relevant prior art is outside of traditional sources, patents can be granted for unoriginal programs. The lack of an easily accessible system for cataloging software patents, for which the USPTO suffered early criticism, further complicated prior art searches.<sup>53</sup> This lack of accessible prior art makes

---

50. See, e.g., Andrew M. Riddles & Brenda Pomerance, *Software Patentee Must Conduct own Search: Prior-Art Searches Made by the Patent Office are not Thorough Enough to be Trusted*, NAT'L L.J., Jan. 26, 1998, at C19. While he does not call for the abolition of software patents, Glynn Lunney does worry that “[i]f courts fail to enforce the nonobviousness requirement and allow an individual to obtain a patent for simply implementing existing methods of doing business through a computer, even where only trivial technical difficulties are presented, entire e-markets might be handed over to patent holders with no concomitant public benefit.” See Glynn S. Lunney, Jr., *E-Obviousness*, 7 MICH. TELECOMM. & TECH. L. REV. 363, 366 (2001); Charles Arthur, *The Patenting of Software is a Complete Mess and Discourages Innovation*, THE INDEPENDENT (London), Jan. 7, 2004, at 9.

51. See, e.g., Simson Garfinkel, *Patently Absurd*, WIRED, July 1994, at 104 (complaining that certain computer applications are taught in first-year programming classes, but judges deciding patent infringement cases are unlikely to know this); Lawrence Lessig, *The Problem with Patents*, THE INDUSTRY STANDARD, Apr. 23, 1999, available at [www.lessig.org/content/standard/0,1902,4296,00.html](http://www.lessig.org/content/standard/0,1902,4296,00.html).

52. Most often, prior art consists of earlier patents. However, non-patent prior art (such as journals, text books, and even Web sites) is important in some fields, including software. See John Allison & Emerson Tiller, *Internet Business Method Patents*, in PATENTS IN THE KNOWLEDGE-BASED ECONOMY 259, 279 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

53. See Garfinkel, *supra* note 51.

it more difficult for patent examiners to determine originality, resulting in too many bad software patents.

¶26 Certainly this complaint contains some truth. Software development does not have as strong a tradition of academic, or even trade press, publishing as some other disciplines, although academic and trade journals in computer science are available.<sup>54</sup> Compared to other fields such as chemistry, software prior art searches are likely to be more difficult for everyone, including would-be inventors.<sup>55</sup> Moreover, since software patents are relatively recent, patent examiners have had little expertise in the area until quite recently. Furthermore, it would be false to claim that every patent issued by the USPTO represents a significant innovation in its field—“bad” patents are likely issued on a regular basis.<sup>56</sup> That being said, banning software patents as the solution to USPTO troubles amounts to throwing out the good with the bad. A more reasonable approach would be to focus on reforming USPTO procedures. In evaluating this argument against software patents, where the anecdotes at first blush sound quite convincing, it is best to keep the broader picture in mind, which we discuss below.

#### a. Anecdotal Evidence Can Be Misleading

¶27 Patent examiners are supposed to grant patents only when an invention is “novel” and “non-obvious.”<sup>57</sup> Novelty is straightforward: as long as an invention has not been created before, novelty is assured.<sup>58</sup> Non-obviousness is a little trickier. The test is whether persons with ordinary skills in the relevant art could have foreseen the invention *at the time* the invention was made.<sup>59</sup> Hindsight does not count here.

¶28 Examiners have always had to balance time spent examining a patent application to ensure its novelty and non-obviousness against the backlog of other patent applications awaiting their review.<sup>60</sup> The sheer volume of patent applications, along with typical job

---

54. E.g., JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH; COMPUTATIONAL LINGUISTICS; CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE; THE COMPUTER JOURNAL ONLINE; COMPUTER GRAPHICS QUARTERLY.

55. Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUDIES 321, 369 (1995).

56. The Internet contains many websites devoted to the numerous “crazy,” “wacky,” “absurd,” or just simply “bad” patents that have been issued. Some example patent titles include: *Non-Lethal Cock Fighting System, Device for Waking People from Sleep, Flying Saucer Submarine*. See, e.g., <http://www.patentoftheweek.com>; <http://www.purdue-law.com/wackypatents.html>; <http://totallyabsurd.com>

57. See 35 U.S.C. § 103 (2004).

58. See 35 U.S.C. §§ 102(a), (e), (g) (2004).

59. 35 U.S.C. § 103.

60. In 2003 alone, about 3,600 USPTO examiners received over 350,000 new patent applications, not counting the patents applied for in applications from earlier years that remained pending and therefore still requiring some attention. United States Patent and Trademark Office, *Mission and Organization of the USPTO*, USPTO PERFORMANCE AND ACCOUNTABILITY REPORT FISCAL YEAR 2003, available at [http://www.uspto.gov/web/offices/com/annual/2003/0401\\_mission\\_org.html](http://www.uspto.gov/web/offices/com/annual/2003/0401_mission_org.html); United States Patent and Trademark Office, *Table 1: Summary of Patent Examining Activities*, USPTO PERFORMANCE AND ACCOUNTABILITY REPORT FISCAL YEAR 2003, available at [http://www.uspto.gov/web/offices/com/annual/2003/060401\\_table1.html](http://www.uspto.gov/web/offices/com/annual/2003/060401_table1.html).

time pressures and USPTO policies that favor granting rather than rejecting patents,<sup>61</sup> implies that the granting of “bad” patents is inevitable. Software patent opponents often point to Amazon’s “one-click” patent as an example.<sup>62</sup>

¶ 29 The critics may or may not be right about Amazon’s one-click patent. A recent analysis of that patent by software historian Martin Campbell-Kelly observes:

Note, however, that the innovation did follow from Amazon.com’s marketing insight that users found the virtual shopping-cart style of ordering [prevalent on Web sites at the time] a turnoff, especially when they were looking for a single item. Such knowledge could have come only from the experience of running an e-commerce Web site, informed by consumer research . . . . To put it another way, if the value of one-click shopping were so obvious, why did no one implement it earlier?<sup>63</sup>

¶ 30 We are neither advocating nor condemning Amazon’s one-click patent here. The point is that triviality, novelty, and non-obviousness must be determined in the proper historical context, not after the fact. Much is obvious after someone else has made it so.

¶ 31 More importantly, a mere recitation of anecdotal evidence of purportedly poor USPTO decisions does not further the debate in any meaningful way. Out of the almost 1.4 million patents in force today in the U.S.,<sup>64</sup> critics can undoubtedly find examples of “bad” patents in any number of disciplines, software included. To truly evaluate software patents we must look beyond individual examples and ask whether, on the whole, the benefits of having software patents outweigh the costs. Answering that question requires careful empirical research.<sup>65</sup>

¶ 32 Given that the patent application procedure is fallible, measures are in place for legally contesting bad patents. Annually, only a few hundred of the roughly one and a

---

61. Patent examiners are compensated based on the number of patents that they either grant or reject. However, because of procedures that allow the postponement of the finality of a rejection, there are strong incentives for the granting of patents. See Robert P. Merges, *As Many as Six Impossible Patents Before Breakfast: Property Rights For Business Concepts and Patent System Reform*, 14 BERKELEY TECH. L.J. 577, 606-09 (1999); see also Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495 (2001).

62. Paul Davidson, *Patents out of Control?*, USA TODAY, Jan. 13, 2004, at 1B.

63. Martin Campbell-Kelly, *Software Patents: A (Wary) Defense*, THE MILKEN INST. REV., Fourth Quarter, at 26, 32 (2003).

64. Japan Patent Office and European Patent Office, *Trilateral Statistical Report* § 2, Graph 2.1 (2002), available at <http://www.uspto.gov/web/tws/tsr2002/ch2/index.html>.

65. To date, there have been very few empirical (as opposed to anecdotal) studies of software patents. Allison & Tiller examined the prior art cited in internet-related patents in comparison with a random sample of more general patents. See Allison & Tiller, *supra* note 52, at 261. Graham and Mowery study the trends in software patenting, particularly among the U.S.-based packaged software development firms. See David Mowery & Stuart Graham, *Intellectual Property Protection in the U.S. Software Industry*, in PATENTS IN THE KNOWLEDGE-BASED ECONOMY 219, 220 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

half million patents in force are contested in court.<sup>66</sup> This statistic suggests—and empirical evidence confirms—that many patents granted are of little economic consequence and therefore not worth litigating.<sup>67</sup> In addition, means other than litigation are available for resolving issues. As a result, most bad patents have little meaningful impact; they are issued and then forgotten.<sup>68</sup> We should not confuse the need for system reforms with the alleged need for abolishing the system entirely. The problem of inevitable bad patents is best addressed by considering reforms in USPTO procedures covering all fields, not just software.<sup>69</sup>

### b. The Process Is Improving

¶33 As discussed above, during the first ten or so years of software patents, pure software patents were not granted. Instead, the software aspects of an innovation were tied to a physical apparatus or to a “process,” such as a way to manufacture a good.<sup>70</sup> Since pure software patents were not granted, the USPTO did not hire examiners with programming backgrounds. For that reason, most of the patent examiners working on software patents were working outside of their field of expertise. Only in the mid-90s did the USPTO begin to hire examiners from software and related fields.<sup>71</sup> While it may be a slow process, over time, the USPTO will develop experience in evaluating software patents that should translate into the granting of fewer “bad” patents.<sup>72</sup>

---

66. Each year, while suits involving around 2,000 patents are typically filed, only about 100 cases a year involving around 125 patents actually make it to trial. See Lemley, *supra* note 61, at 1501; Mark A. Lemley, *Essay: Reconceiving Patents in the Age of Venture Capital*, 4 J. SMALL & EMERGING BUS. L. 137, 142 (2000). However, Arti Rai argues that even if obvious patents are eventually struck down by the courts as invalid, “such litigation is likely to be expensive and time-consuming.” See Arti Rai, *Part II: Judicial Issue: Addressing the Patent Gold Rush: The Role of Deference to PTO Patent Denials*, 2 WASH. U. J.L. & POL’Y 199, 212 (2000). Thus, Rai contends that companies obtaining bad (or invalid) patents can nonetheless obtain a “first-mover advantage” in fast-paced industries, and therefore the litigation safety valve has less meaning. *Id.* Countering Rai’s argument to some extent, Jean Lanjouw and Mark Schankerman find that “advantages in settlement are exercised quickly, before extensive legal proceedings consume both court and firm resources.” See Jean O. Lanjouw & Mark Schankerman, *Enforcement of Patent Rights in the United States*, in PATENTS IN THE KNOWLEDGE-BASED ECONOMY 145, 145-46 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

67. See Lemley, *supra* note 61, at 1511-17. Furthermore, research by Mark Lemley indicates that most litigation is not over software patents, but instead is over low-tech mechanical patents. Lemley, *supra* note 66, at 141.

68. See Lemley, *supra* note 61, at 1506. According to USPTO data, from 1999 to 2003 about one-sixth of the patents up for renewal were left to expire. In that period, over 260,000 patents expired because of non-renewal. See Table 1: *Summary of Patent Examining Activities*, *supra* note 60. Little more than one-third of all U.S. patents are maintained throughout the entire twenty-year period. See Lemley, *supra* note 60, at Table 1.

69. See generally Nancy T. Gallini, *The Economics of Patents: Lessons from Recent U.S. Patent Reform*, 16 J. ECON. PERSPS. 131 (2002) (discussing the recent patent reform literature).

70. See MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 107 (2003).

71. Julie Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 11-12, n.39 (2001).

72. Merges notes that similar complaints were made of the USPTO when biotechnology patents were new. See Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope*, 90 COLUM. L. REV. 839, 905-06 (1990).

¶ 34 Not surprisingly, the U.S. patent classification system was not set up with software patents in mind, making prior art searches more difficult.<sup>73</sup> Moreover, relevant prior art for software cannot always be found in prior patents or in academic and trade journals. Nonetheless, better techniques for finding unconventional prior art are developing and will continue to improve. In fact, the USPTO unveiled an initiative in 2000 that aimed to improve existing processes for examining business method and software patents.<sup>74</sup> One of the stipulations was improved access to prior art databases; another required more rigorous non-patent prior art and foreign prior art searches.<sup>75</sup> The current efforts for improvement offer hope for further self-improvement.

### c. Empirical Evidence Suggests Some Criticisms May Be Unfounded

¶ 35 The handful of academic empirical studies that examine software patents do not paint nearly as dismal a picture as the ad hoc stories do.<sup>76</sup> For example, John Allison and Emerson Tiller compare general patents with Internet business method patents, using the number of prior art references listed in a patent as a measure of the rigor of the prior art search.<sup>77</sup> Surprisingly, for patents issued between 1996 and 1998, they find that the Internet business method patents<sup>78</sup> and general patents have statistically insignificant differences in patent references. Not surprisingly, the Internet business method patents contain more references to non-patent prior art. This was also true of the subcategory of Internet software technique patents. Since there are much fewer software patents in existence, one might suspect that the additional non-patent prior art references were out of necessity, substituting for patent references. At least for the narrow time period that they examine, however, prior art counts that do not consider the quality of the prior art cited suggest that prior art searches for software patents may meet non-software patent standards.

## 2. Patents can hold standards hostage

¶ 36 The second procedural complaint against software patents centers on their role in industry standard setting. Many industries have a certain set of standards that ensure a

---

73. Cohen & Lemley, *supra* note 71, at 12-13.

74. Mowery & Graham, *supra* note 65, at 229.

75. *Id.*

76. For examples of complaints regarding software patents' prior art deficiencies, see Greg Aharonian, *17,500 Software Patents To Issue in 1998*, INTERNET PATENT NEWS SERV., Oct. 18, 1998 (as cited in Merges, *supra* note 61, at 589 n.31).

77. See Allison & Tiller, *supra* note 52, at 261-62.

78. Software patents are not synonymous with business method patents. Software patents and business method patents are related, most obviously because the legal reasoning behind granting both kinds is similar. However, important differences in important areas exist. For example, the Amazon.com one-click purchase patent is fundamentally a business method patent: novelty and non-obviousness were argued and granted on commercial innovation grounds, not on technical innovation. See Chris Morton, *Business-Method Patents: Of Questionable Validity?* 6 COMPUTER L. REV. & TECH. J. 321, 324-25 (2002). Many legitimate arguments against business method patents can be used against software patents in the general sense, but acknowledging the very real differences between the two is important when discussing their relative merits.

degree of interoperability and compatibility of products. Industries that display network effects, such as the computing industry, have a special need for such standards.<sup>79</sup> In the computing industry, components from many different producers need to work together. Thus, strong standards help preserve interoperability and competition. In the hardware industry, standards are important for compatibility between various components, such as sound or video cards and hard drives.<sup>80</sup> Standards are just as important for the software industry, where applications need to interface with at least the operating system, and often with each other as well.<sup>81</sup> Some critics have claimed that when standards involve patented technology, the standard can be held hostage because a single company has power over something that is relied on by many, perhaps thousands, of other companies.<sup>82</sup>

¶ 37 Particularly problematic are so-called hidden or submarine patents. For example, a company secretly navigates the patent application process for a computing format while that format gains wide acceptance as a de facto standard, only to reveal the patent once the format is well ensconced. Software patent opponents point to the patent on the LZW compression algorithm as an example.<sup>83</sup> In 1984, the LZW algorithm was described in a trade magazine by one of the algorithm's authors.<sup>84</sup> Unisys held the rights to the algorithm and patented the method described, but this fact was not disclosed in the article. LZW came to be used in many file archiving programs, such as the popular Unix program "compress." In 1987, CompuServe published specifications for the "GIF" image format, which used LZW. GIF eventually became extremely popular and was included in several standards, such as W3C web standards. In the mid-1990s—a decade after disclosing the LZW technology in a public article—Unisys asserted its patent rights by deciding to charge license fees. Predictably, the move angered the many developers and companies who were accustomed to free use of the standard.<sup>85</sup>

¶ 38 Just as in the first criticism that poor software patents are issued, much of this complaint rings true. But again, the extreme measure of eliminating software patents to solve the standards problem is not warranted. Other means for preventing patent holders from hijacking standards are available and preferable. We discuss these below.

---

79. See Jason V. Morgan, *Open Source Software and Software Patents: Finding the Common Ground in a Patent Pool 3* (2002) (unpublished B.S. thesis, University of Utah School of Computing), available at <http://opensource.mit.edu/papers/morgan.pdf>.

80. For example, the Video Electronics Standards Association (VESA) is a standard-setting organization that specializes in video and display standards.

81. The Internet, for example, relies on HTTP standards for interoperability between clients' web-browsers and the web-servers for the websites.

82. See David Berlind, *Internet Standards must be Royalty Free*, ZDNET, Apr. 12, 2002, at [www.techupdate.zdnet.com/techupdate/stories/main/0,14179,2861033,00.html](http://www.techupdate.zdnet.com/techupdate/stories/main/0,14179,2861033,00.html); Free Software Foundation, *FSF's Position on W3 Consortium "Royalty-Free" Patent Policy*, at <http://www.gnu.org/philosophy/w3c-patent.html> (last updated Mar. 9, 2004).

83. See, e.g., Free Software Foundation, *Why There Are No GIF Files on GNU Web Pages*, available at <http://www.gnu.org/philosophy/gif.html> (last updated Feb. 9, 2004).

84. Terry A. Welch, *A Technique for High-Performance Data Compression*, 17 IEEE COMPUTER 8, 15-19 (1984).

85. Frank Lenk, *Innocent GIF File Stirs Dispute: Debate on the Internet Demonstrates a New Interaction between Consumer, Corporation*, THE GLOBE & MAIL, Mar. 14, 1995, at C2.

### a. Standard setting organizations can help

¶ 39 One such solution lies with standard-setting groups themselves. While complete protection against hidden patents by unrelated third parties is difficult, standard-setting bodies can ensure that member companies and participants in standard-setting procedures commit to full disclosure and reasonable royalties on any related intellectual property. Tough contractual disclosure obligations as the price for participation in the standard process can help to protect a standard's accessibility after wide-scale adoption. Mark Lemley examined the rules that standard-setting bodies can establish to prevent members from asserting their patent rights and argues that existing case law generally implies strong enforceability of such rules.<sup>86</sup>

¶ 40 Some standard-setting bodies rely on a licensing principle known as "RAND," where necessary components of a standard have to be licensed on terms that are "Reasonable And Non-Discriminatory."<sup>87</sup> Partly out of fear of any appearance of price fixing, which risks investigation and prosecution by the antitrust authorities as well as treble-damage private law suits in the U.S., standard setters traditionally settle for vague guidelines for what constitutes "reasonable."<sup>88</sup> After a standard has been adopted, what could be considered reasonable royalties of an essential component to a standard is quite different from what could be considered reasonable ex ante, when the component is still competing with other technologies. Potential antitrust concerns are legitimate, but the pro-competitive effects of establishing reasonable royalties prior to setting a standard appear well worth the risk. Antitrust regulators could alleviate problems associated with standard setting and patents by encouraging more definitive royalty terms between standard setters or at least a process for determining those terms that would not get the standard setting bodies in an antitrust mess.

¶ 41 Even if standards groups choose not to institute formal intellectual property rules, they can do a better job of researching property rights. Greater diligence in patent searches by standard setters could have prevented the Unisys/LZW problem—after all, the patent had been granted long before the GIF format's release. Prior to settling on a standard, organizations have considerably more bargaining power to extract reasonable licensing terms from patent holders in exchange for using the patented technology in an endorsed standard. Therefore, careful research ex ante could go a long way in lessening the danger of companies exerting patent rights ex post. Standards bodies can also

---

86. See Mark A. Lemley, *Intellectual Property Rights and Standard-Setting Organizations*, 90 CAL. L. REV. 1889, 1918-21 (2002).

87. See, e.g., Internet Engineering Task Force, *The Internet Standards Process – Revision 3*, § 10.3.2(C) (Oct. 1996), available at <http://www.ietf.org/rfc/rfc2026.txt>. See also Carl Shapiro, *Navigating the Patent Thicket: Cross Licenses, Patent Pools and Standard Setting*, INNOVATION POL'Y AND THE ECON, 130 (Adam Jaffe et al. eds., 2001).

88. Josh Lerner and Jean Tirole note that "[t]here is now widespread agreement among policymakers and economists that patent pools may benefit both intellectual property owners and consumers, provided that the pools include patents that are complementary or blocking [as opposed to substitutes]." JOSH LERNER & JEAN TIROLE, EFFICIENT PATENT POOLS 2 (Nat'l Bureau of Econ. Research, Working Paper No. 9175, 2002). This was not always the case. Washington was quite hostile towards patent pools just after World War II. See *Id.*

institute rules that punish firms attempting to hide patents during the standards review process. Such situations are not limited to the software industry. In the mid-1990s, Dell decided to enforce its patent rights on the “VL-bus” standard, after it had certified to VESA (a standard setting organization of which Dell was a member) that it held no relevant intellectual property. Dell was eventually forced to surrender its patent rights in a settlement after charges of anticompetitive conduct from the Federal Trade Commission (FTC).<sup>89</sup>

### b. USPTO reforms should lessen the problem

¶42 Recent reforms at the patent office require patents to be published eighteen months after the date of application.<sup>90</sup> While loopholes in the law may allow some patent applicants to avoid the eighteen-month publication rule, this new rule should help to reduce submarine patents.<sup>91</sup>

¶43 Other possibilities include adopting a pre-grant challenge system similar to that of the Japanese and European Patent Offices. A pre-grant challenge would allow interested third parties to present information during the patent prosecution process to challenge the validity of a proposed patent. This process could lift some of the research burden off of the USPTO and it would be a cheaper and possibly more effective means of challenging patents than presently exists in the U.S. On the other hand, pre-grant challenges have the potential for dragging out the patent process and would be open to abuse by competitors attempting to block rivals’ patents.<sup>92</sup>

---

89. See Shapiro, *supra* note 87, at 124.

90. 35 U.S.C. § 122(b)(1)(A) (2004). See also United States Patent and Trademark Office, *Eighteen-Month Publication of Patent Applications*, available at <http://www.uspto.gov/web/offices/pac/pgpub.htm> (last modified Nov. 16, 2003).

91. In an empirical study, Daniel Johnson and David Popp analyze the impact of the new publication rule. They conclude that information diffusion does indeed begin with patent publication. See Daniel Johnson and David Popp, *Forced Out of the Closet: The Impact of the American Inventors Protection Act on the Timing of Patent Disclosure*, 34 RAND J. ECON. 96 (2003). As a result, they argue that early disclosure will be beneficial to the research community in general and should increase the pace of innovation, although it may harm some inventors who would have preferred the longer period of secrecy. *Id.* at 111. Johnson and Popp find that the issue of submarine patents is economically irrelevant. Despite their prominence in the intellectual property rights debate, submarine patents represent only .00028% of the more than two million patents granted between 1971 and 1993. *Id.* at 99-99. Thus, while the American Inventors Protection Act will certainly make indefinitely long submarine patents impossible, Johnson and Popp feel they were a minor problem to begin with. Mann argues that the information dissemination benefit of patents is unlikely to apply to the software industry: “As others have noted, with software cases the Federal Circuit has interpreted the disclosure requirement in such a way as to minimize the likelihood that disclosures will require information that is directly useful to competing inventors. Moreover, given the rapid pace of innovation, it will not often be the case that information disclosure in a patent application filed years earlier will be of immediate value to competitors in the industry.” RONALD J. MANN, *THE MYTH OF THE SOFTWARE PATENT THICKET: AN EMPIRICAL INVESTIGATION OF THE RELATIONSHIP BETWEEN INTELLECTUAL PROPERTY AND INNOVATION IN SOFTWARE FIRMS 50-1* (Univ. of Texas Sch. of Law, Law & Econ. Working Paper No. 022, 2004).

92. See, e.g., Federal Trade Commission, *To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy* 8 (2003), available at

## B. Economic Theory and the Justification of Patents

¶44 Another class of complaints against software patents is rooted in economic theory. The classic economic justification for patents lies in their ability to spur innovation.<sup>93</sup> Because intellectual property is easily copied once it is created and disclosed, but often difficult to develop initially, potential inventors may be reluctant to invest in research for fear that they will not be able to recoup their investments after their invention is unveiled. By providing inventors a temporary right to exclude others from using their invention, patents allow inventors to appropriate the returns to their investment and thus provide incentives both to create more innovations and to make those innovations public. Software patent critics claim that this traditional argument does not apply to software creation, thus software patents are not justified.

### 1. Software was innovative before patenting

¶45 As discussed above, while a handful of software patents snuck through in the 1970s, software could not be patented as such until the 1981 *Diehr* decision.<sup>94</sup> Even then, software was only patentable if disguised as hardware until a 1994 decision removed that barrier.<sup>95</sup> Today, U.S. courts officially recognize software patents, but they still represent a small subset of overall patenting.<sup>96</sup> Software patent critics note that the computing industry has been one of the most innovative and productive industries in history, even long before patents entered the innovation equation. Software patent opponents thus argue that patents are unnecessary because the industry has proven its ability to innovate without the help of patents.<sup>97</sup> If it is not broken, why fix it?

¶46 This argument glosses over two important facts. Software has gone through several phases over time; a brief look at each phase helps to clarify whether the available mechanisms for intellectual property protection were adequate spurs to innovation. Further, patents cannot be considered in isolation. For-profit companies developing proprietary software will attempt to protect their investments with all available means. Alternative protections, such as trade secrecy or copyright, involve costs for society just as patents do. We expand on both of these points below.

---

<http://www.ftc.gov/os/2003/10/innovationrptsummary.pdf>; Jay Kesan, *Carrots and Sticks to Create a Better Patent System*, 17 BERKELEY TECH. L.J. 763, 778 (2002).

93. For a seminal work, see Kenneth J. Arrow, *Economic Welfare and the Allocation of Resources for Invention*, in *THE RATE AND DIRECTION OF INVENTIVE ACTIVITY: ECONOMIC AND SOCIAL FACTORS* 619 (Richard R. Nelson ed., 1962).

94. See Campbell-Kelly, *supra* note 63, at 31; *Diamond v. Diehr*, 450 U.S. 175 (1981).

95. *In re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994).

96. There is no official definition of “software patents,” but by taking Graham & Mowery’s definition, software patents comprised 3.8% of all U.S. patents in force in 1997. Graham & Mowery, *supra* note 65, at 232.

97. See, e.g., JAMES BESSEN & ERIC MASKIN, *SEQUENTIAL INNOVATION, PATENTS, AND IMITATION* 2 (M.I.T. Dep’t of Econs., Working Paper No. 00-01, 2000).

### a. The industry has changed

¶47 The software industry today is very different from and in many ways not comparable to the industry during its early years.<sup>98</sup> In the 1960s, hardware manufacturers financed software production. The margins on expensive proprietary hardware were high enough that custom software often was considered an add-on.<sup>99</sup> Today, the few segments of the hardware market that continue to provide robust margins are insufficient to sponsor software development for the entire market, especially for software developed for commodity hardware.<sup>100</sup> As hardware became a more widespread product, available to consumers as well as businesses, margins in the hardware industry fell, and an enormous market for software on personal computers was created. Demand for expensive customized software is too low to support a wide customized software industry. In contrast, demand for software that can run on general purpose computers in one's home has grown exponentially in the past two decades.<sup>101</sup>

¶48 Programs today are also much more complex than they were in the infancy of the computer industry. Decades of innovation in software and increasingly powerful hardware have raised the bar for what is possible and what is marketable.<sup>102</sup> Widespread commodity hardware has decreased the viability of an entire industry based on the vertically integrated systems model.<sup>103</sup> Since the old business model for software production is no longer practical, it is unwise to extrapolate from one industry era to another.

¶49 Court decisions in the mid-1990s, and in particular one regarding an infringement suit brought against Borland by Lotus, narrowed the previously strong intellectual property protections afforded to software by copyright law.<sup>104</sup> Patents gained importance as copyrights lost their effectiveness. An empirical study by Mowery and Graham suggests that patent protection could be a substitute for copyright protection rather than a complement.<sup>105</sup> The authors find that large firms are registering their software copyrights less frequently than in earlier years. That is, just as patent protection becomes more readily available for software, companies appear to be reducing their reliance on copyright. While copyright is conferred automatically, only copyrights that are officially registered within five years of original publication are presumed valid under law.<sup>106</sup> Moreover, copyright holders cannot sue for infringement until a copyright is

---

98. Except where otherwise noted, this brief history of the software industry is based on CAMPBELL-KELLY, *supra* note 70.

99. *See id.* at 6.

100. *See id.* at 87.

101. *See id.* at 31-66.

102. *See id.* at 1-27.

103. *See Mowery & Graham, supra* note 65, at 221.

104. Mowery & Graham, *supra* note 65, at 225-26; Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, *affirmed by an equally divided court*, 516 U.S. 233 (1996).

105. Mowery & Graham, *supra* note 65, at 248-51. Lemley and O'Brien also argue that patents and copyright are substitutes. *See Mark A. Lemley & David W. O'Brien, Encouraging Software Reuse*, 49 STAN. L. REV. 255, 299-304 (1997).

106. 17 U.S.C. § 410 (2004).

registered.<sup>107</sup> These factors, among others, provide incentives for companies to register copyrights.

¶ 50 In contrast, Ronald Mann, suggests another reason for the decreased copyright registrations. While the Copyright Office requires the deposit of only portions of the registered source code, companies still worry about reverse-engineering.<sup>108</sup> Evidence from Mann's recent survey also suggests that despite the reduced protection copyright affords software, it still has a fundamental role to play due to the differences in these two types of IP protection. Copyright protects only a program's "expression," not its functionality.<sup>109</sup> As any software developer will acknowledge, programs can be written in any number of ways and still achieve equivalent functionality. Thus, copyright can be important for guarding against piracy among users, even though it affords little protection against copying by competitors.<sup>110</sup> Patents, on the other hand, do protect against copying by competitors because they cover functionality. One of Mann's interviewees sums up the point nicely:

Copyright solves one problem, which is the whole or partial copying of an expressive application. The whole or partial copying of an application by a pirate you can get. But it doesn't really protect us from sharing our technical information broadly and a company then understanding how our products work. Patents are inter-industry mechanisms for creating value. Copyright is creating protection between the industry and the channel or end customers.<sup>111</sup>

¶ 51 As a final observation on the "need" for patent protection in the software industry, bear in mind that innovation is not something that either does or does not happen: it happens either more or less. Asserting that the industry was innovative and productive in the past does not prove that the rate of innovation and investment cannot be improved today, or even that the rate could not have been improved had patents existed in the past.<sup>112</sup>

## 2. Software development is a sequential and cumulative process

---

107. 17 U.S.C. § 411 (2004).

108. See Ronald J. Mann, *Secured Credit and Software Licensing*, 85 CORNELL L. REV. 134, 148-49 (1999).

109. See *Baker v. Selden*, 101 U.S. 99, 103-05 (1879). For a discussion, see MANN, *supra* note 91, at 19-20.

110. MANN, *supra* note 91, at 19-20.

111. *Id.* at 20.

112. The empirical literature on whether patents increase or aid innovation has yet to yield any conclusive evidence. As Bronwyn Hall observes, "casual observation suggests that business method patents are not being used to provide innovation incentives as much as they are being used to extract rents *ex post*, but this evidence could be misleading. We do not know whether there would have been as much entry into internet businesses or new financial offerings in the absence of the patent system." See BRONWYN H. HALL, BUSINESS METHOD PATENTS, INNOVATION, AND POLICY 11 (Nat'l Bureau of Econ. Research, Working Paper No. 9717, 2003). For a thorough survey of the literature, see Adam Jaffe, *The U.S. Patent System in Transition: Policy Innovation and the Innovation Process*, 29 RES. POL'Y 531, 536-49 (2000).

### making patent thickets likely

¶ 52 The theoretical economics literature argues that when innovations are sequential and cumulative, patents may impose more than the typical exclusion-period costs.<sup>113</sup> For instance, a patent for an invention early in the innovative process could impose a toll on each sequential innovation that relies on it. Subsequent inventors, therefore, face higher transaction costs—they must pay licensing fees before they can further refine a technology. As the tolls build during the technology’s development path, later research could be discouraged altogether. Alluding to the famous argument for property rights, over-patenting has been dubbed the “tragedy of the anticommons,” as too many people with exclusionary rights can cause underutilization of resources.<sup>114</sup> Shapiro uses another metaphor: the patent thicket.<sup>115</sup>

¶ 53 Software patent critics observe that the necessary conditions which underlie these theoretical arguments are present in today’s high technology industry.<sup>116</sup> Every complex program is made up of hundreds, if not thousands, of smaller pieces of code.<sup>117</sup> The novelty in a program lies either in the manner in which these pieces are combined, or in an additional component or algorithm that is developed by the software creator. Software is a cumulative form of engineering, in that new programs rely heavily on old software, or at least on ideas obtained from old software. The novel portion of a program might just be a small part of the whole code base.<sup>118</sup>

¶ 54 Based on the theoretical literature, software patent opponents claim that patent thickets will develop in the software industry. Thus, they assert that patents allow a company or individual to prevent the type of incremental innovation that is so important in the software industry. When small pieces of software that are adaptable to a multitude of applications can be and are patented, it becomes increasingly likely that each complex program will infringe someone’s patent. Therefore, opponents argue that developers have incentives to “over-patent” for strategic or defensive reasons in order to gain leverage in cross-licensing negotiations.<sup>119</sup> Strategic patenting results in a patent thicket,

---

113. Carl Shapiro, *Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard Setting*, in INNOVATION POLICY AND THE ECONOMY 119, 122-23 (Adam Jaffe et al. eds., 2001); Suzanne Scotchmer, *Standing on the Shoulders of Giants: Cumulative Research and the Patent Law*, 5 J. OF ECON. PERSP. 29, 32-35 (1991).

114. Michael Heller and Rebecca Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, SCIENCE, May 1, 1998, at 698.

115. Shapiro, *supra* note 113, at 119.

116. See, e.g., BESSEN & MASKIN, *supra* note 97, at 2-3.

117. Windows, an example at the top end of the scale, is made up of tens of millions of lines of code. Robert L. Mitchell, *Why Windows Should Think Small*, COMPUTERWORLD, Aug. 25, 2003, at 37. See also Paul Krill, *Chasing Bugs Away: Bugs beware! Best practices, software tools, and code-inspection services are on the prowl*, INFOWORLD, Oct. 24, 2003, available at [www.infoworld.com/article/03/10/24/42FEbugs\\_1.html](http://www.infoworld.com/article/03/10/24/42FEbugs_1.html)

118. See John S. Liebovitz, *Inventing a Nonexclusive Patent System*, 111 YALE L.J. 2251, 2284-85 (2002).

119. JAMES BESSEN & ROBERT M. HUNT, AN EMPIRICAL LOOK AT SOFTWARE PATENTS 47, Table 1 (Research on Innovation, Working Paper No. 03-17/R, 2004). For a critique of their approach and results,

an impenetrable barrier to further innovation. Moreover, opponents claim that patents give larger companies disproportionate power in the industry and overwhelm whatever positive innovation effects patents might have.<sup>120</sup>

¶ 55 Clearly, software development is a sequential and cumulative process. Thus, a software anticommons is a theoretical possibility. However, a more relevant question to ask is whether a software anticommons is a reality. Other industries with longstanding histories of patenting could be categorized as having cumulative and sequential R&D, yet they do not display signs of innovation gridlock. Accordingly, it would seem that patent-induced anticommons are the proper subject of an empirical question, not a theoretical one. We examine the available evidence below.

#### a. Patent thickets are not inevitable

¶ 56 Walsh, Arora, and Cohen looked into whether excessive and defensive patenting has led to an anticommons tragedy in the biomedical industry.<sup>121</sup> The biomedical industry displays many of the preconditions for such an anticommons, including an increase in the patenting of upstream research tools and the existence of patenting for strictly defensive purposes. The authors surveyed researchers in the field on whether the current state of patents in the industry was preventing promising projects from being undertaken. Walsh et al. found little evidence of an actual anticommons breakdown in the industry. Instead, they found that licensing of patented technology was common and that research exemptions were used liberally and advantageously, such that biomedical patents rarely prevented further innovation in either the private or the public sectors. They note that only a small number of overall patents granted end up having a real impact on any given potential research project.<sup>122</sup>

¶ 57 Ziedonis and Hall took a similar approach to the semiconductor industry.<sup>123</sup> As in both the software and biotech industries, the semiconductor industry has demonstrated rapid technological advancement in a cumulative R&D environment. The authors conducted a survey of industry participants and complimented that with patenting data on semiconductor firms from 1979-1995. The interviews and econometric analysis suggest two results. First, strategic patenting does exist and is a primary cause for the increased propensity to patent over the last two decades. Second, however, “vertical disintegration” of the industry is a positive effect of stronger patent rights. That is, patents helped “facilitate entry by specialized firms,” which are firms that focus on semiconductor design, do not have integrated manufacturing capabilities, and rely more

---

see ROBERT W. HAHN & SCOTT WALLSTEN, A REVIEW OF BESSEN AND HUNT’S ANALYSIS OF SOFTWARE PATENTS (AEI-Brookings Joint Center, Working Paper, Nov. 2003).

120. Danny Bradbury, *Canadian Innovation Choked by U.S. Laws*, NAT’L POST, Nov. 17, 2003, at FE01; see Yager, *supra* note 30.

121. John P. Walsh, Ashish Arora, & Wesley M. Cohen, *Effects of Research Tool Patenting and Licensing and Biomedical Innovation*, in PATENTS IN THE KNOWLEDGE-BASED ECONOMY 285 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

122. *Id.* at 286, 331-36.

123. Bronwyn Hall & Rosemarie Ziedonis, *The Patent Paradox Revisited: An Empirical Study of Patenting in the U.S. Semiconductor Industry, 1979-1995*, 32 RAND J. ECON. 101 (2001).

heavily on patents for appropriation of R&D returns.<sup>124</sup>

¶ 58 Neither of these related industry studies proves the appropriateness or inappropriateness of patent rights for software. In a recent working paper, however, Mann presents similar evidence for the software industry.<sup>125</sup> He surveyed approximately fifty industry executives among software development firms, venture capitalists, and financiers. When Mann raised the idea of patent thickets in his interviews, it was “universally” rejected.<sup>126</sup> Instead, interviewees noted that the industry’s large patent portfolio holders (notably IBM and Microsoft) had policies of freely licensing their IP to all legitimate users.<sup>127</sup> They also pointed out that large companies had little to gain except for bad publicity from pressing their patent advantage against smaller players, since start-ups do not have the funds to pay large settlements.<sup>128</sup>

¶ 59 While none of these studies are definitive (as they are largely based on survey evidence), they do demonstrate that the many theoretically possible negative effects of patents on innovation do not translate into black and white results when analyzed empirically. Studies such as these raise new questions, questions that have yet to be properly examined for the software industry. Considerable empirical work must be done before patents are vilified as inevitably leading to non-negotiable thickets. The final cost-benefit calculus for patents will depend on the empirical evidence.

#### **b. There are other ways to prevent patent thickets**

¶ 60 Defensive patenting may be inevitable, but it need not devolve into patent thickets that lead to licensee hold-up and project abandonment. Mechanisms such as cross-licensing and patent pools are effective ways to share technology.<sup>129</sup> Moreover, both of these practices appear to be common in the computer industry.<sup>130</sup>

¶ 61 Policy shifts could increase the accessibility of licensing solutions. Carl Shapiro argues that one problem is that the FTC has been overzealous and misguided in

124. *Id.* at 125. The authors are careful to note that strategic patenting behavior is not an inherent feature of strong patent rights, in that as a patent thicket grows, it should be increasingly difficult to pass non-triviality tests. Thus, the descent into impenetrable patent deadlock is not infinite.

125. MANN, *supra* note 91, at 58-9.

126. *Id.* at 53.

127. Policies listed on the websites for these two companies corroborate the interviews. *See, e.g.*, IBM, *IBM Worldwide Patent Licensing Practices*, available at <http://www.ibm.com/ibm/licensing/patents/practices.shtml> (last viewed 5/18/04); Microsoft, *Microsoft Announces Expanded Access to Extensive Intellectual Property Portfolio* (Dec. 3, 2003), available at <http://www.microsoft.com/presspass/press/2003/dec03/12-03ExpandIPPR.asp>. *See also* CNET.com, *Patents: a Necessary Evil* (Jan. 5, 2002), at <http://news.com.com/2009-1001-801896.html>.

128. However, interviewees also noted that IBM would be unwilling to grant a license to any party that refused to grant IBM parallel access to its IP, implying that patents can be used to pressure smaller companies to open access to their IP. MANN, *supra* note 91, at 54. Of course, this too weakens the argument that patent thickets prevent innovations, as it indicates increased licensing rather than blocking.

129. Shapiro, *supra* note 113, at 122.

130. *See* Seth Shulman, *Software Patents Tangle the Web*, 103 MIT TECH. REV. 68 (2000).

preventing pro-competitive patent pools.<sup>131</sup> Antitrust regulators have taken hard stances on complementary patent pools, in which patents from different companies that are necessary to make one product are pooled in a simple licensing scheme.<sup>132</sup> This idea is tied into standard setting as well. Without antitrust concerns to overshadow the process, more pro-competitive licensing could occur. Lerner and Tirole suggest that enforcers could rely on the presence of independent licensing (where patent pool members can offer licenses individually, outside the pool) as a screening device for welfare-enhancing pools.<sup>133</sup>

### 3. Patents can reduce the software “commons”

¶62 According to open-source software developers and users, the biggest problem is that patented software reduces the software “commons.”<sup>134</sup> In other words, the existence of software patents means more restrictions on what is includable in open-source software. Open-source proponents argue that eliminating restrictions on all code would mean more free software for everybody. The argument resembles one commonly leveled at the pharmaceuticals industry: if only we did away with drug patents, we could give people more life-saving medicine at a marginal cost.<sup>135</sup>

¶63 On the surface, this argument holds some appeal. The fatal flaw lies in the argument’s static nature: future innovation is ignored. Whether eliminating software patents would lead to more freely available software depends critically on the future production of software. If patents help to increase the production of software, then eliminating them could actually decrease the “commons” in the long run.

#### a. Future innovation must be considered when measuring the software “commons”

¶64 Patents do not fully protect all of the information embodied in an innovation. One theory, espoused by Polk Wagner, divides information related to an innovation into three types: the core information, the derivative information, and the information that was inspired by the innovation.<sup>136</sup> For example, a newly developed hybrid corn species embodies information on the hybrid corn itself (the core), better hybrid corn compared to earlier varieties (the derivative), and information on the hybridization process (the inspired). Patents grant extensive control over the core innovation, but only partial

131. Shapiro, *supra* note 87, at 137-8, 147.

132. See JOSH LERNER, JEAN TIROLE & MARCIN STROJWAS, COOPERATIVE MARKETING AGREEMENTS BETWEEN COMPETITORS: EVIDENCE FROM PATENT POOLS 4 (Nat’l Bureau of Econ. Research, Working Paper No. 9680, 2002).

133. For suggested antitrust authority guidelines on dealing with patent pools, see LERNER & TIROLE, *supra* note 88, at 37-39.

134. See, e.g., Free Software Foundation, *Patent Reform is Not Enough*, available at <http://www.fsf.org/philosophy/patent-reform-is-not-enough.html> (last updated Feb. 22, 2004); LAWRENCE LESSIG, THE FUTURE OF IDEAS 214-15 (2001).

135. See, e.g., Mark Weisbrot, *A Prescription for Scandal*, BALT. SUN, Mar. 21, 2001, at 17A.

136. R. Polk Wagner, *Information Wants To Be Free: Intellectual Property and the Mythologies of Control*, 103 COLUM. L. REV. 995, 999 (2003).

control over derivative innovations, and no control over merely inspired innovations. If a positive relationship exists between degree of control and production of innovation, Wagner suggests that increased control can lead to an increase in total information (the sum of the three types).<sup>137</sup> The steps are as follows: increased control over core information provides incentives to innovate while the core information embodied in the new innovation is protected by a patent (and therefore not part of the commons); thus, the derivative and inspired information increases the commons today. Furthermore, once the patent expires, its core information is added to the commons as well. As a result, the incentives for future innovation that a system of control can spur may dominate the effect of releasing all information today through eliminating patents. While complications (e.g., whether patents block future innovation under a sequential innovation model) could mitigate Wagner's findings, his main point is an important one. Opening code today affects code production tomorrow.

¶65 The answer to the question of how to maximize innovation and information dissemination does not by itself fully address the usefulness of patents. If patents do spur innovation, this value must still be weighed against the real cost from the temporary exclusion that patents grant. Yet before we get to that question, we must first determine the effect of patents on innovation, a question for which definitive answers have yet to be seen.<sup>138</sup>

#### **b. The alternative to patents is worse**

¶66 One important issue that is often ignored in the software patent debate is realistic alternatives to the patent regime. The FSF dream in which programmers abandon the profit motive and freely develop and share all the software that users could ever want will never be more than just a dream. While “free software” proponents would prefer the end of proprietary software, eliminating software patents will not bring about this change. Without patents, the industry would move back to complete reliance on copyrights and trade secrets, both of which could be much worse for the industry and open source.<sup>139</sup> In terms of information disclosure and a software commons, copyrights on programs and trade secrets on the code lead to less disclosure overall by corporations as compared to patents.

¶67 Eliminating patents may also harm small companies. Some research indicates that patents are important to the early stage financing of small businesses. Josh Lerner studied the importance of patents to the acquisition of venture capital financing and found that patent scope had a positive affect on the ability of a company to acquire venture funding.<sup>140</sup> Mann and Sager's findings indicate that patents positively influence the

---

137. *Id.* at 1023.

138. As noted *supra* note 112.

139. Posner and Landes argue that “patent law combats this incentive [to keep an innovation as a trade secret]” by requiring disclosure. See WILLIAM M. LANDES & RICHARD A. POSNER, *THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW* 294 (2003).

140. Joshua Lerner, *The Importance of Patent Scope: An Empirical Analysis*, 25 *RAND J. ECON.* 319, 320 (1994).

number of financing rounds that a small firm receives.<sup>141</sup> Software patents also may provide small businesses with negotiation leverage against large companies that they would not have in a world of software copyrights and trade secrets.<sup>142</sup> At the same time, the literature suggests a number of negative consequences for small firms caused by patents, including barriers to entry, legal strong-arming by firms with large patent portfolios, and general chilling effects on research.<sup>143</sup> In the final analysis, it is likely that patents are a double-edged sword: they can both help and hurt small businesses, depending on the circumstances.

¶ 68 Controversial effects on small businesses aside, patents may facilitate a secondary market in reusable software components.<sup>144</sup> There have always been people who have sold software components, and current marketplaces such as ComponentSource—which boasts sponsorship from the likes of Microsoft and Sun—are evidence of a recent upswing in component-based software development. While copyright may be sufficient for some developers, patents can encourage the licensing of software that would otherwise remain hidden.<sup>145</sup>

#### IV. CONCLUSION

¶ 69 Open-source software advocates have advanced a number of arguments against software patents over the last few years. Their attacks range from the procedural (the USPTO has done a poor job of issuing software patents) to the theoretical (software development is a sequential process, so allowing patents will lead to an innovation-stifling thicket of intellectual property rights).

¶ 70 Certainly, some of these arguments hold a kernel of truth. The USPTO is not perfect and bad patents are undoubtedly issued in a number of fields. Software development does indeed meet the threshold criteria for the potential of patent thickets in that innovations are sequential and cumulative. However, the next link in the open-source chain of logic, which requires the abolition of software patents, does not follow. For some of the complaints, far more sensible alternatives are available. For example, many scholars have written on the need to reform the patent office and have offered concrete suggestions for doing so.<sup>146</sup> General reform of the application and granting process would improve patents in numerous fields, not just software. Moreover, to keep software standards accessible, even in the face of patented technology, standard-setting

---

141. MANN, *supra* note 91, at 38 n.192 (citing Mann & Sager forthcoming 2004).

142. See generally Paul Heckel, *Debunking the Software Patent Myths*, 35 COMMS. OF THE ACM 121 (1992); Mark Radcliffe, *Patents: A Small Price to Pay for Progress*, CIO MAG., Aug. 1, 2003, available at [www.cio.com/archive/080103/debate\\_pro.html](http://www.cio.com/archive/080103/debate_pro.html).

143. Paul Davidson, *Patents Out of Control?*, USA TODAY, Jan. 13, 2004, at B1; Peter Williams, *Small Developers Fear EU Software Patents*, COMPUTING, Sept. 1, 2003, available at [www.computing.co.uk/news/1143293](http://www.computing.co.uk/news/1143293).

144. Lemley & O'Brien, *supra* note 105, at 299, 304.

145. Michael Guntersdorfer & David Kay, *How Software Patents can Support COTS Component Business*, IEEE SOFTWARE, at [www.computer.org/software/homepage/2002/03COTS/](http://www.computer.org/software/homepage/2002/03COTS/) (last viewed May 18, 2004); MANN, *supra* note 91, at 43.

146. See, e.g., Merges, *supra* note 61, at 591; Rai, *supra* note 66, at 218.

organizations can take practical steps like employing well-designed contracts to limit the chance that members will assert intellectual property rights after a technology is solidly entrenched.

¶ 71 Evidence does not support the argument that patents simply are not needed to spur innovation, or that patents block future innovation by causing non-negotiable thickets of rights and reducing the software commons. Arguments like these are based largely on theory with a smattering of anecdotes thrown in, rather than on hard empirical evidence. Despite the long history of IP literature, the empirical evidence linking patents and innovation is still inconclusive for all fields, let alone software. The theoretical literature on this issue is equally ambiguous. It would be a dangerous precedent to ban software patents based merely on the possibility that the truth resides in one strain of the theoretical literature arguing against patents. Similarly, while the law and economics literature has established that patent thickets are possible, what little empirical evidence is available on this point indicates that they are not inevitable. Means of working around the exclusionary effects of patents—such as patent pools and cross-licensing—are available to the software industry.

¶ 72 Considering the evidence in its totality, the call for abolishing software patents appears far too drastic a step. Admittedly, problems do exist, but the literature also contains examples of the positive effects of patents, ranging from increased information dissemination to improved venture funding for small firms. In the end, reform is far more attractive than abolition, because it retains the good while minimizing the bad.