View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Brooklyn Law School: BrooklynWorks

Brooklyn Law Review

Volume 78 | Issue 3 Article 4

2013

# Software Patents and/or Software Development

Wendy Seltzer

Follow this and additional works at: https://brooklynworks.brooklaw.edu/blr

Recommended Citation

Wendy Seltzer, *Software Patents and/or Software Development*, 78 Brook. L. Rev. (2013).
Available at: https://brooklynworks.brooklaw.edu/blr/vol78/iss3/4

This Article is brought to you for free and open access by the Law Journals at BrooklynWorks. It has been accepted for inclusion in Brooklyn Law Review by an authorized editor of BrooklynWorks.

www.manaraa.com

# Software Patents and/or Software Development*

*Wendy Seltzer*†

## INTRODUCTION

Many contemporary treatments of the patent system begin with Fritz Machlup's damning with faint praise:

> If we did not have a patent system, it would be irresponsible, on the basis of our present knowledge of its economic consequences, to recommend instituting one. But since we have had a patent system for a long time, it would be irresponsible, on the basis of our present knowledge, to recommend abolishing it.[1]

Yet he concludes that for all its imperfections, the patent system is still worth keeping.[2] Patent may introduce costs and inefficiencies, this analysis goes, but since patents serve a necessary function in creating incentives to innovate, we must bear and mitigate their costs. The time is ripe to revisit that analysis.

In the case of software patents, I challenge the incentive side of the equation: Patents do not provide a useful incentive to innovate in the software industry, I contend, because the patent promise ill-suits the engineering and development practices and business strategies of software production. The problem is not merely an inefficiency in implementation of software patent, but a structural mismatch between where the

---

* © 2013 Wendy Seltzer. This work is licensed under a Creative Commons Attribution 3.0 Unported License.

† Senior Fellow, Yale Law School Information Society Project, and Senior Researcher, Berkman Center for Internet & Society at Harvard University. Thanks to workshop participants at the University of Colorado, Princeton Center for Information Technology Policy, Yale Law School, and TPRC. Research performed while a Fellow at Silicon Flatirons Center at University of Colorado School of Law was funded by a generous grant from Brad Feld to the Silicon Flatirons Center. All opinions are those of the author. Contact wendy@seltzer.org.

[1] SUBCOMM. ON PATENTS, TRADEMARKS, & COPYRIGHTS OF THE S. COMM. ON THE JUDICIARY, 85TH CONG., AN ECONOMIC REVIEW OF THE PATENT SYSTEM 80 (Comm. Print 1958), [hereinafter S. SUBCOMM., ECONOMIC REVIEW].

[2] *Id.*



www.manaraa.com

incentive applies and how software innovation happens. Even an ideally implemented software patent—well examined, fully disclosed and enabling, and properly scoped in light of the prior art—would fail to serve the incentive functions intended by the Constitution, the Patent Act, and standard patent theory.

Previous scholarship, whether critical or congratulatory of software patent, has largely failed to examine the structure of software development and the institutional specifics of patent's operation in this industry. I therefore look at these mechanics: How is the incentive function of patent believed to operate? How does it operate in the software industry? Does the tool serve its goals? Addressed head on, even before compounding the issue with side effects and unintended consequences, I conclude that the answer to this last question is "no." Present knowledge and experience now offer sufficient evidence that patents disserve software innovation.

Part I situates the problem by providing an account of the tangle of patent lawsuits, licenses, and threats in the mobile phone industry.

Part II describes the nature of software development, its sources of innovation, and its business environment. This part draws on sources from engineering, computer science, and business and strategy literature, as well as the experiences of commercial and open source software developers. I discuss several ways in which software development differs from the canonical model of manufacturing widgets as well as the challenges of going from idea to implementation, including prototyping, revising to meet user needs, and debugging. With the aim of identifying common frameworks, this part focuses on the nuts and bolts of how systems function, a feature shared with New Institutional Economics literature. While market dynamics differ among segments, we can identify commonalities derived from the underlying nature of software.

Part III reviews existing legal theories of patent incentives and innovation. It formalizes the mismatch between incentive theory and software patent practice. In many of the accounts that attribute value to software patents, a circularity exists: startups claim that patents are important because investors demand them, whereas venture capitalists, who view patents as a signal of capacity or uniqueness, are in fact seeing a show aimed at attracting investment rather than a demonstration of genuine novelty or value to the customer market. Kitch's prospect theory does no better to validate

www.manaraa.com

software patents. The patent claim is staked too early to give the proprietor a useful coordinating or notice function.

Part IV applies the theories about patent incentives and innovation to software more specifically. Where do software developers and venture capital backers seek patents, and how do individuals and firms use them? Looking particularly at the timing of patent's intervention in the system, I conclude that it encourages idea-claiming, not innovation; idea-generation, but not implementation, debugging, and deployment. A player focused on patenting can obtain numerous patents without developing any of the technologies to useful levels of deployment or disclosure, leaving a minefield of abstract patent claims for others who actually deploy software. Hence, the "troll" problem is particularly acute in the software field. Because generating a patentable idea for an initial invention rarely creates a bottleneck in the software development process, software patents that never reach implementation more frequently create entangling thickets than productive incentives. Here, I also analyze the patent alternatives that are available to protect software development: trade secrecy, copyright, first-mover advantage, and market complements.

Part V uses this analysis to reflect on the institutional dynamics of patent law. In particular, close analysis of software patents and software development adds to the ongoing debate over technology—specificity versus uniformity in law and the proper role of courts versus Congress in deciding patent questions. This analysis will also assist in better framing the question of how best to uphold the constitutional mandate to "promote the progress of science and useful arts."[3]

## I. THE MOBILE PHONE MESS

### *A. Smartphone Patent Wars*

Smartphones are everywhere. As the hottest selling consumer product category in consumer electronics history, smartphones have provoked moral panics (are we losing ourselves behind screens, neglecting interpersonal communication? are kids "sexting"?); safety risks (don't text and drive); and development optimism (in developing countries, where more people have cell phones than landlines, the phone is becoming the basis for mobile commerce and access to computing power).

[3] U.S. CONST. art. I, § 8, cl. 8.

www.manaraa.com

These devices have been at the heart of the last few years' most ferocious patent storms.[4] Throughout 2012, in the United States alone, dozens of patent litigations focused directly or indirectly on mobile phone technology.[5] Some of those suits relate to hardware features but more frequently concern the smartphone's software capabilities.[6] For example, Lodsys, a Marshall, Texas corporation, has no known products, but the company offers patented technologies "available for licensing"[7] and has sued or threatened to sue dozens of application software companies alleging that in-app purchases and rating functions utilize Lodsys proprietary technology.[8] Although Lodsys claims that Apple, Google, and Microsoft have patent licenses covering their own "nameplate" products,[9] it argues that those do not extend

[4] *See, e.g.*, Apple, Inc. v. Samsung Electronics Co., 11-cv-1846 (N.D. Cal. 2012); Apple Inc. v. Motorola Mobility, No. 11-cv-178 (W.D. Wisc. 2012); Microsoft Corp. v. Motorola, Inc., No. C10-1823 (W.D. Wash. 2012); In the Matter of Mobile Electronic Devices, including Wireless Communication Devices, Portable Music and Data Processing Devices, and Tablet Computer, 2012 WL 4077563 (U.S. Int'l Trade Comm'n).

[5] *See generally* Fred I. Williams & Rehan M. Safiullah, *The Smartphone Patent Wars: A U.S. Perspective*, 18 IP LITIGATOR, July/Aug. 2012, *available at* http://cdn.akingump.com/images/content/5/5/v2/5506/IPLIT070812WilliamsSafiullah.pdf.

[6] *See generally id.* As the case of software-defined radio demonstrates, the lines between hardware and software are changing. *See, e.g.*, articles discussed in *Software Defined Radio*, ARRL.ORG, http://www.arrl.org/software-defined-radio (last visited Mar. 22, 2013); *see also* Stephen M. Blust, *Software Based Radio*, *in* SOFTWARE DEFINED RADIO: ENABLING TECHNOLOGIES 5 (Walter H.W. Tuttlebee, ed. (2002)) (describing a "shift from employing a traditional hardware-focused application-specific approach to radio implementation to using a software application to perform the radio tasks on a computing platform").

[7] *Licensing*, LODSYS GROUP LLC, http://www.lodsys.com/licensing.html (last visited Oct. 2, 2012).

[8] Eric Mack, *Mobile Patent Wars: A Closer Look at How Everyone Loses*, PC WORLD (Nov. 6, 2011, 9:00 PM), http://www.pcworld.com/article/239873/mobile_patent_wars_a_closer_look_at_how_everyone_loses.html.

[9] *See Q: Lodsys Is Trying to Force Apple to Take a License by Pressuring IOS Developers*, LODSYS GROUP LLC (May 15, 2011), http://www.lodsys.com/1/post/2011/05/q-lodsys-is-trying-to-force-apple-to-take-a-license-by-pressuring-ios-developers.html; *Q: What About Other Operating Systems such as Android?*, LODSYS GROUP LLC (May 15, 2011), http://www.lodsys.com/1/post/2011/05/q-what-about-other-operating-systems-such-as-android.html.

www.manaraa.com

to third-party developers.[10] Apple has moved to intervene,[11] and Google has called for reexamination of the patents.[12]

Patents are clearly costly.[13] Their drafting and prosecution take time and money that could be spent on product development. Litigation costs start at nearly half a million dollars before a case even gets to trial.[14] Damages in the event of a loss can run to millions of dollars.[15] Further, companies are now making acquisitions with a primary aim to obtain patents. A coalition including Apple, Microsoft, and Research In Motion paid $4.5 billion to acquire Nortel's patent portfolio in the company's bankruptcy auction in July 2011.[16] These patents amounted to $700,000 apiece for their coalition of purchasers—who, given that the sale included no going concern, acquired only the use of the patents but none of the know-how or experience of the inventors. When Google announced its agreement a few weeks later to acquire Motorola Mobility Inc. for $12.5 billion, the acquisition of a major mobile hardware manufacturer was widely read as a purchase of a defensive portfolio of mobile software patents as a means to

---

[10] *Apple's License Claim Disputed*, LODSYS GROUP LLC (May 31, 2011), http://www.lodsys.com/1/post/2011/05/-apples-license-claim-disputed.html; *see also* Sarah Perez, *Patent Holding Firm Lodsys Goes After Android Developer for Use of In-App Payments*, READWRITEWEB (May 27, 2011), http://www.nytimes.com/external/readwriteweb/2011/05/27/27readwriteweb-patent-holding-firm-lodsys-goes-after-andro-98683.html.

[11] Apple Inc.'s Motion to Intervene, Lodsys, LLC v. Combay, Inc. et al., 11-cv-272, (E.D. Tex. June 9, 2011 *available at* http://www.scribd.com/doc/57508610/0-Apple-Motion-to-Intervene-Against-Lodsys.

[12] *See Google Steps Up to Defend Android Developers from Patent Lawsuit*, WIRED.COM (Aug. 13, 2011, 2:20 AM), http://www.wired.com/gadgetlab/2011/08/google-android-lodsys-patent.

[13] *See generally* JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 39-42 (2008). "Innovators can benefit from patents and at the same time be burdened with dispute costs that exceed the value of those benefits." *Id.* at 96.

[14] STEVEN M. AUVIL & DAVID A. DIVINE, AM. INTELLECTUAL PROP. LAW ASS'N, REPORT OF THE ECONOMIC SURVEY 2011, at 35 (2011).

[15] A recent PWC study found that annual median damages awards ranged from $2.4M to $10.5M. *See* CHRIS BARRY ET AL., THE CONTINUED EVOLUTION OF PATENT DAMAGES LAW (PriceWaterhouseCoopers ed., 2010).

[16] *See* Chris V. Nicholson, *Apple and Microsoft Beat Google for Nortel Patents*, N.Y. TIMES DEALBOOK (July 1, 2011, 4:58 AM), http://dealbook.nytimes.com/2011/07/01/apple-and-microsoft-beat-google-for-nortel-patents/ (last updated July 1, 2011, 8:31 PM). Similarly, in 2011, another consortium consisting of Microsoft, Oracle, Apple, and EMC spent $450 million for 882 patents owned by Novell. *See CPTN Holdings LLC and Novell Inc. Change Deal in Order to Address Department of Justice's Open Source Concerns*, DEP'T OF JUSTICE (Apr. 20, 2011), http://www.justice.gov/opa/pr/2011/April/11-at-491.html; Press Release, Novell Completes Merger with Attachmate and Patent Sale to CPTN Holdings LLC, Novell (Apr. 27, 2011), http://www.novell.com/news/press/2011/4/novell-completes-merger-with-attachmate-and-patent-sale-to-cptn-holdings-llc.html.

www.manaraa.com

protect Google's Android mobile operating system and its users.[17] From a strategic perspective, there were plenty of reasons for Google not to enter the hardware business, but it apparently became clear that without patents to counter-assert against patent attackers, Android would lose the confidence of other hardware makers and fail.

Big companies are not only on the defensive, though; most are also wielding patents on offense in the smartphone arena. Apple has stopped smartphone imports at the border, suing before the International Trade Commission (ITC) to enjoin HTC, Samsung, and Motorola from bringing allegedly infringing smartphones into the United States.[18] In Europe, Samsung sued to block Apple's sales.[19] Microsoft has sued Motorola, among others; Nokia has sued Apple.[20] And the list goes on.

### *B. The Idea of Pinch-to-Zoom*

Patents are supposed to promote innovation. Even these costs might be justified, their proponents say, if they result in more innovation. To understand patent basics, we might look at a patent from the middle of the road: 7,479,949, "Touch screen device, method, and graphical user interface for determining commands by applying heuristics,"[21] and 7,812,828, "Ellipse Fitting for Multi-Touch Surfaces," issued October 12, 2010.[22] These patents belong to Apple, an extraordinarily successful company that manufactures actual hardware and software, as compared to non-practicing entities like Lodsys who do neither. Apple Inc. has sued and been sued on infringement claims for both of these patents.

The '949 patent claims the use of touch gestures on a screen to issue commands by three different methods: as a stored computer *program* (or software); as a *device* operated by the software; and finally, as a *method* implemented within the

---

[17] *See* Amir Efrati & Spencer E. Ante, *Google's $12.5 Billion Gamble*, WALL ST. J., Aug. 16, 2011, at A1 ("The Motorola deal also gives the search giant a trove of more than 17,000 patents to defend itself against a rash of lawsuits against its Android software . . . .").

[18] *See* Williams & Safiullah, *supra* note 5, at 7-9.

[19] Lance Whitney, *Apple Suing to Block Samsung Phone Sales in Japan*, CNET NEWS (Sept. 8, 2011, 6:25 AM), http://news.cnet.com/8301-13578_3-20103189-38/apple-suing-to-block-samsung-phone-sales-in-japan/.

[20] Williams & Safiullah, *supra* note 5, at 6-7.

[21] U.S. Patent No. 7,479,949 (filed Apr. 11, 2008), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "7479949").

[22] U.S. Patent No. 7,812,828 (filed Feb. 22, 2007), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "7812828").

www.manaraa.com

software.[23] The patent describes this program/device/method in more than 350 pages with 293 drawings, yet it includes no source code and no demonstration of the underlying implementation.[24] Anyone who wanted to implement Jobs et al.'s "method, and graphical user interface for determining commands by applying heuristics"[25] would still have to write his or her own code.

Many aspects of the user interface are described in the specification and diagrams of the '949 patent, but after being narrowed during prosecution and reexamination, the patent claims only some aspects of scrolling in one or two dimensions and selecting which areas of a screen to move.[26] But because the patent is not limited to any particular program used by Apple, its narrowed claims expand again upon the company's actions enforcing it. Any device could be accused of infringing the patent if the device permits finger-driven scrolling and panning; if it determines, based on touch motion, the degree of vertical-only scrolling; or if it permits a two-fingered twist to rotate. Many devices are so accused.[27]

True, these gestures have been used to great effect on the iPhone, which Steve Jobs introduced in January 2007 with a slide reading "Patented"[28] and released to the public (to camped-out lines in front of Apple stores) on June 29, 2007.[29] But market demand does not necessarily indicate that the idea was a novel or non-obvious solution, the legal prerequisites for the grant of any patent.[30] Rather, it indicates that Apple

---

[23] '949 Patent, at [54].

[24] '949 Patent.

[25] *See id.*

[26] *See generally id.*

[27] *See* Complaint at 4, Apple, Inc. v. Motorola, Inc., No. 11-08540, 2011 WL 6257973 (Dist. Ct. N.D. Ill. 2012) (accusing a number of Motorola mobile devices of infringing U.S. Patent No. 7,479,949, such as the "Droid, Droid 2, Droid X, Cliq, Cliq XT, BackFlip, Devour A555, Devour i1, and Charm").

[28] *See* Nilay Patel, *Apple vs. Palm: The In-Depth Analysis*, ENGADGET (Jan. 28, 2009, 1:28 PM), http://www.engadget.com/2009/01/28/apple-vs-palm-the-in-depth-analysis/; *Macworld 2007 Keynote Photo Gallery*, ENGADGET, http://www.engadget.com/photos/macworld-2007-keynote-photo-gallery/131095/ (last visited Aug. 25, 2012).

[29] *See* Jeremy W. Peters, *Gave Up Sleep and Maybe a First-Born, but at Least I Have an iPhone*, N.Y. TIMES, June 30, 2007, at C3, *available at* http://www.nytimes.com/2007/06/29/technology/29cnd-phone.html.

[30] *See* 35 U.S.C. §§ 102, 103 (2006) (setting forth conditions for novelty and non-obvious subject matter). Novelty and utility are prerequisites of patentability. *Id.* Thus, the patent must not have been disclosed to the public (by various means) prior to application. *Id.* § 102. Additionally, a patent may not be granted if the invention was a non-obvious solution, that is, the idea must not have been obvious to a person skilled in the art. *Id.* § 103. In other words, it must not have been obvious to other potential inventors to "combine references," or ideas contained in research available to others at

www.manaraa.com

implemented its solution well. Use of that "invention" is on its own clearly insufficient to make a product succeed. For example, HP/Palm—which Apple accused of copying its multitouch gestures in Palm's WebOS—shuttered the entire WebOS product line in response to low sales, despite critical acclaim for innovative features distinct from Apple's.[31]

Apple's release of the iPhone certainly inspired the more rapid development of other full-featured smartphones with touch screens. According to Steven Levy, Google dropped a keyboard-focused model from its Android efforts after seeing the iPhone.[32] In 2008, Steve Jobs visited Google, where he "felt that Google had stolen Apple's intellectual property to do so, appropriating features for which Apple had current or pending patents."[33] According to Levy, Jobs

> apparently was successful, at least in the first version of the Google phone, in halting [Google's] implementation of some of the multitouch gestures that Apple had pioneered. . . . According to one insider, Jobs demanded that Google remove support of those gestures from Android phones. Google complied, even though those gestures, which allowed users to resize images, were tremendously useful for viewing web pages on handheld devices.[34]

Yet Android had multitouch support in its software, though it remained hidden; the code was present but unused.[35] Outside the United States, and in modified U.S. phones, users could "pinch-to-zoom" with impunity.[36] In mid-2010, when

---

the time, in order to make the invention. *Id.*; *see also* KSR Int'l Co. v. Teleflex Inc., 550 U.S. 398, 407 (2007).

[31] *See* Brian X. Chen, *In Flop of H.P. TouchPad, an Object Lesson for the Tech Sector*, N.Y. TIMES (Jan. 1, 2012), http://www.nytimes.com/2012/01/02/technology/hewlett-packards-touchpad-was-built-on-flawed-software-some-say.html (describing the WebOS platform); Mat Honan, *HP Killing WebOS Devices*, GIZMODO (Aug. 18, 2011, 3:20 PM), http://gizmodo.com/5832291/hp-killing-webos (describing HP's announcement that it was ceasing development).

[32] *See* STEVEN LEVY, IN THE PLEX: HOW GOOGLE THINKS, WORKS, AND SHAPES OUR LIVES 221 (2011); *see also* Scott Cleland, *What Really Made Steve Jobs So Angry at Google?*, GIZMODO (Sept. 10, 2012), http://gizmodo.com/5941817/what-really-made-steve-jobs-so-angry-about-google.

[33] LEVY, *supra* note 32, at 221.

[34] *Id.*

[35] *See* John Herrman, *The G1's Deepest, Darkest Secret: Hidden Multi-Touch*, GIZMODO (Nov. 18, 2005, 5:28 AM), http://gizmodo.com/5091705/the-g1s-deepest-darkest-secret-hidden-multitouch (discussing the ability of Google's first Android smartphone, the T-Mobile G1, to detect multiple, simultaneous touches after code in the touchscreen's software driver that had been "commented out" was re-enabled); *see also* Ryan Gardner, *Proving the G1 Screen Can Handle Multi-Touch*, RYEBRYE (Nov. 17, 2008), http://www.ryebrye.com/blog/2008/11/17/proving-the-g1-screen-can-handle-multi-touch.

[36] *See* Frederic Lardinois, *Nexus One Gets Official Multitouch Support (Updated)*, READWRITEWEB (Feb. 2, 2010), http://www.readwriteweb.com/archives/nexus_one_

www.manaraa.com

Google's Eric Schmidt was no longer on Apple's board, Apple sued HTC before the ITC and federal district court in Delaware, alleging patent infringement.[37] "Within days, Google rolled out a change in Android's operating system: it would now support the pinch and stretch multitouch gestures that Jobs had demanded Google remove."[38] Google's response to an inquiry from *ReadWriteWeb* invoked standards:

> Looking around the mobile industry, from Apple to Palm to HTC and Motorola—it's clear that pinch-to-zoom technology has become a standard and popular way for users to interact with their mobile phones. Likewise, Android users can now truly benefit from this capability with the availability of Android 2.1, which powers a new class of devices with larger touchscreens and more interactive features.

Despite the rationalizations of "increased processor power," it appears that patent threats retarded the functionality of Android's touch screen by almost two years.[39]

Google is still feeling the impact of Apple's patent claims in lawsuits against its Android partners.[40] Larry Page himself emphasized the patent angle of its merger with Motorola on Google's corporate blog:

> We recently explained how companies including Microsoft and Apple are banding together in anti-competitive patent attacks on Android. The U.S. Department of Justice had to intervene in the results of one recent patent auction to "protect competition and innovation in the open source software community" and it is currently looking into the results of the Nortel auction. Our acquisition of Motorola will increase competition by strengthening Google's patent portfolio,

---

gets_official_multitouch_support.php ("Oddly enough, Nexus One (and Motorola Droid) users outside of the U.S. have always been able to make use of these multitouch features by default.").

[37] Complaint, Apple Inc. v. High Tech Computer Corp., Nos. 10-166-GMS, 10-167-GMS, 2011 WL 124446 (D. Del. Jan. 14, 2011) (No. 10CV00166), 2010 WL 976411 (filed Mar. 2, 2010); Complaint, Apple Inc. et al v. High Tech Computer Corp., Nos. 10166-GMS, 10-167-GMS, 2011 WL 124446 (D. Del. Jan. 14, 2011) (No. 10CV00167), 2010 WL 711393 (filed Mar. 2, 2010).

[38] *See* LEVY, *supra* note 32, at 237.

[39] Jacqui Cheng, *T-Mobile, Google Finally Unveil the First Android Phone*, ARSTECHNICA (Sept. 23, 2008, 10:56 AM), http://arstechnica.com/gadgets/2008/09/t-mobile-google-finally-unveil-the-first-android-phone/.

[40] At least until the Motorola deal, Google did not make hardware, but developed the operating system and released it open source, as well as licensed applications for use by partners. Google depends on independent hardware vendors to make smartphones, which it may help support with marketing. Google does not charge for the operating system, but earns revenue by advertising to the people who use Google's services on their phones. *See* Brian X. Chen & Claire Cain Miller, *Android Is No. 1, But Google Says It Still Makes Little Money*, N.Y. TIMES BITS BLOG (Jan. 20, 2012, 6:13 PM), http://bits.blogs.nytimes.com/2012/01/20/android-small-revenues/.

www.manaraa.com

> which will enable us to better protect Android from anti-competitive threats from Microsoft, Apple and other companies.[41]

Android users already faced disruption to their devices from several patent lawsuits, and after Google's opponents purchased Nortel's patent portfolio, Google and its Android partners (including HTC and Motorola) had reason to fear a deepening thicket. Without many patents of its own, Google could not have made the traditional counterstrike of suing its attackers for infringement.[42] Motorola's mobile portfolio (17,000 issued patents and 7,500 pending applications) added to the Android's arsenal.[43]

Of course Motorola also makes hardware—smartphones that run Android—but few analysts emphasized that point.[44] There, the acquisition raises strategic questions for Google: Can it convincingly offer the Android platform to companies against whom it now competes? Even if Google maintains Motorola as a separate business, as Page says the company intends,[45] will now-competing vendors such as HTC, Samsung, and Acer be reassured of Google+Motorola's neutrality among them?

Owning a handset maker could improve Android, if it shortens the feedback loop for problem-reporting and new ideas, but it could hurt the platform—and its end-users—even more by scaring off competing hardware vendors, which would shrink the base to which new applications are written and reduce the diversity of options available to end-users. As proprietor of an open, multi-sided market, Google needs to serve Android's hardware vendors, app developers, and end-users well enough that a large enough group of each continue to bring it value while directing end-users to the ads whose sale puts money into Google's pocket.

The patent motivations are more straightforward in the Motorola purchase. It does not take deliberate copying to infringe a patent, and patents are granted on small enough

---

[41] *Supercharging Android: Google to Acquire Motorola Mobility*, GOOGLE: OFFICIAL BLOG (Aug. 15, 2011), http://googleblog.blogspot.com/2011/08/supercharging-android-google-to-acquire.html.

[42] *Id.*

[43] Stephen Shankland, *Google's Page Explains Motorola Acquisition*, CNET (Aug. 15, 2011), http://news.cnet.com/8301-30685_3-20092367-264/googles-page-explains-motorola-acquisition/.

[44] Dan Rowinski, *The New Motorola: Google's Hardware Division Steps into the Future*, READWRITEWEB (Sept. 6, 2012), http://www.readwriteweb.com/mobile/2012/09/the-new-motorola-googles-hardware-division-steps-into-the-future.php.

[45] *Google to Acquire Motorola Mobility*, GOOGLE INVESTOR RELATIONS (Aug. 15, 2011), http://investor.google.com/releases/2011/0815.html.

www.manaraa.com

increments of software advances that an independently developed application may incorporate dozens to hundreds of elements on which others claim patents.[46] At millions of dollars a lawsuit, it is expensive to disprove them.[47] At least if those other innovators are also making phones or software, Google is now more likely to hold patents on what they are doing too, paving the way for a potential cross-license rather than a lawsuit.[48]

Shouldn't we be able to avoid those patent threats and cross-licensing transaction costs? As Google's pre-Motorola travails showed, it is almost impossible to opt out of the patent system, even if one chooses to publish and not patent his or her inventions.[49] By contrast, the copyright system permits one to share work under a permissive license such as a Creative Commons CC-BY,[50] while defense against infringement requires only proof that one has never accessed another's work.[51] Patent instead puts the developer on the defensive: one can truly guarantee patent-safety only by ensuring that every bit of technology used was published more than seventeen to twenty years ago.[52] Even then, the developer faces the threat of suit for infringing a non-obvious combination of the prior art.[53]

Patent is dis-serving its purpose in this market. If a single device has 250,000 infringement opportunities,[54] the

---

[46] Steve Lohr, *A Patent War in Your Pocket*, N.Y. TIMES, Aug. 25, 2012, at A4, *available at* http://www.nytimes.com/2012/08/26/technology/apple-samsung-case-shows-smartphone-as-lawsuit-magnet.html.

[47] *See* BARRY, *supra* note 15.

[48] Ted Sichelman & Stuart J.H. Graham, *Patenting by Entrepreneurs: An Empirical Study*, 17 MICH. TELECOMM. & TECH. L. REV. 111, 121 (2010).

[49] Timothy B. Lee, *Top Judge: Ditching Software Patents a "Bad Solution,"* ARSTECHNICA (May 13, 2012) (quoting Former Federal Circuit Judge Paul Michel, an expert in the field of intellectual property, as arguing that "software firms don't have the option to opt out of the patent system"). Rare but perhaps not impossible: Richard Hipp of SQLite says he only uses twenty-year old, published algorithms to keep his code free of patent clouds. *See Msg #00026*, OSDIR.COM (Aug. 2009), http://osdir.com/ml/sqlite-dev/2009-08/msg00026.html (last visited Oct. 31, 2012).

[50] *See* CREATIVE COMMONS, http://creativecommons.org (last visited Oct. 31, 2012).

[51] *See* Baxter v. MCA, Inc., 812 F.2d 421, 423 (9th Cir. 1987) ("[A] plaintiff may establish [copyright infringement of software] by circumstantial evidence of: (1) defendant's access to the copyrighted work prior to the creation of defendant's work, and (2) substantial similarity of both general ideas and expression between the copyrighted work and the defendant's work").

[52] 35 U.S.C.A. § 154(a)(2) & (c)(1) (West 2012).

[53] *See supra* note 30 (discussing the requirements for patentability).

[54] *See* Richard Waters, *Patent Hunting is Latest Game in Tech Bubble Circuit*, FIN. TIMES (Inside Business) (July 27, 2011, 7:37 PM), http://www.ft.com/cms/s/0/16025f76-b868-11e0-b62b-00144feabdc0.html#axzz25guuQvQz; Richard Waters, *Tech Patent Arms War Reaches New Level of Intensity*, FIN. TIMES, (Inside Business) (Mar. 30, 2011, 5:50 PM), http://www.ft.com/intl/cms/s/0/b0da8540-5aea-11e0-a290-00144feab49a.html#axzz25guuQvQz.

www.manaraa.com

system puts almost all the power on the side of those who would hold up, rather than advance, progress in the useful arts.[55] Apple, Google, HTC, and others in the smartphone market are making products in spite of, not because of, a patent system in which they are more or less unwilling participants.

### *C. Is Apple's Success Attributable to the Patent Incentive?*

No doubt the iPhone has been terrifically successful, and some of that success is likely due to its interface design. That does not, however, answer two important questions about the patent incentive: First, did the *patent* incentive spur the iPhone development, or would Apple and its investors have gone down the same road if they could have made money by selling iPhones without the assurance that they could sue those who introduced similar features into phone interfaces? Second, do the incentive's benefits outweigh the harm to competition?

To the extent that Apple has created a visual/gestural *language* for smartphone control, we might question whether exclusive ownership is even appropriate. Much of the value of a touch standard has been created by the public, who use it, and by application developers who build on its foundation.[56]

---

[55] *See* Michael A. Heller & Rebecca S. Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, 280 SCIENCE 698, 698 (1998).

[56] *See* Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CALIF. L. REV. 479, 483 (1998). Lemley and McGowan describe the typical "network effects" scenario as one in which "the utility that a user derives from consumption of a good increases with the number of other agents consuming the good" and that "a network effect exists where purchasers find a good more valuable as additional purchasers buy the same good." *Id.* So, to the extent that more mobile phone creators could adopt Apple's touchscreen technology, consumers would gain more value, first, as more users interacted with one another by means of this technology and, second, as more application developers created more programs for use over this platform. Based on these network effects, the value of the iPhone then derives in part from widespread adoption of phones using this technology, regardless of whether Apple or Samsung manufactured any particular device. One could easily imagine a software company that determined that a touchscreen-based application could be profitable only if 50% of mobile phones in the market could support the software. If the iPhone had 35% market share but patents on touch-screen technology prevented other companies from adopting this technological standard, consumers would never see that new application because it would never be profitable for that developer to bring it to market. But, even if one other mobile phone manufacturer were permitted to adopt the technology, assuming no patent prohibitions for doing so, purchase of the new phone could drive the market adoption of touch-screen phones over 50% and thereby create the incentives for that developer—and likely many others—to create a new touch-screen application. The availability of new applications, in turn, would make the iPhone more valuable and could even drive its market share beyond its original 35% market saturation. Joseph Farrell & Phillip Weiser trace the theoretical underpinnings of the often-conflicting doctrines of *internalizing complementary efficiencies* (ICE) and modularity. The theory of ICE, underlying much of antitrust law, maintains that "a monopolist has incentives

www.manaraa.com

Consumers' software choices are often shaped by real or virtual network effects.[57] In seeking those network effects, they may be choosing based less on intrinsic qualities of the product[58] than on its ability to interoperate with those already in the market. A possibility arises, then, that many of the returns on a software product may owe less to superior product quality or moral desert on the part of its creators than to a bandwagon around de facto standards.[59]

The '949 patent itself was one of a line developed out of work at FingerWorks, a company purchased by Apple in 2005.[60] Yet at the time the patent was written, the functions it described were neither novel nor non-obvious—the legal prerequisites to any patent grant.[61] Its novelty is scant against the background of a great deal of research in human-computer interaction. There are indubitably real advances in the state of the art, but they come from many areas—academic research,

---

to provide access to its platform when it is efficient to do so." Joseph Farrell & Philip J. Weiser, *Modularity, Vertical Integration, and Open Access Policies: Towards a Convergence of Antitrust and Regulation in the Internet Age*, 17 HARV. J.L. & TECH 85, 88 (2003). By contrast, modularity reflects the idea that allowing competitors free access to a platform (e.g., a touchscreen interface) spurs innovation, increases market entry, and lowers prices. *Id.* at 95. Given that a patent grants a monopoly over an invention, then, some might suggest that we properly encourage vendors to produce standards by letting them capture all the benefits of the standard, including permitting the monopolist to license its technology to others when it decides that it is efficient to do so. Conversely, by giving them too large an exclusion right, critics of granting patents on software may argue that a patent could diminish the public value of the common standard through the loss of positive externalities gained through network effects.

[57] *See* CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY 174-75 (1998).

[58] *See, e.g.*, Pritam Pal, *HP's WebOS Tablet the Touchpad—The Good, the Bad, and the Ugly*, GIZMOWATCH (July 15, 2011), http://www.gizmowatch.com/entry/hps-webos-tablet-touchpad-good-bad-ugly/ (indicating that the quality of the WebOS tablet might match or exceed that of iOS).

[59] Some have argued that not all network effects are created equal. For example, even though Android devices hold top market share, iOS devices are nonetheless winning the platform war because software developers develop for iOS first and for Android second. The availability of more and better applications—a consequence of developer network effects—is what matters in mobile platform competition, the result of which may ultimately relegate market share and intrinsic design to secondary considerations. *See* Pascal-Emmanuel Gobry, *Understanding the "Network Effect" in the Mobile Platform War*, BUS. INSIDER (Mar. 5, 2012), http://articles.businessinsider.com/2012-03-05/research/31123118_1_ios-android-platform.

[60] *See* Juliana Reyes, *How I Sold My Company to Apple: Jeff White, Former FingerWorks CEO [Q&A]*, TECHNICALLY PHILLY (Jan. 9, 2013, 10:30 AM), http://technical.ly/philly/2013/01/09/jeff-white-fingerworks-apple-touchscreen/ (discussing the work of FingerWorks cofounders John Elias and Wayne Westerman).

[61] *See generally* Bill Buxton, *Multi-Touch Systems that I Have Known and Loved*, BILLBUXTON.COM, http://www.billbuxton.com/multitouchOverview.html (last updated March 2, 2012); *see also* Timothy B. Lee, *If Android Is a "Stolen Product," then so Was the iPhone*, ARSTECHNICA (Feb. 23, 2012, 12:00 PM), http://arstechnica.com/tech-policy/2012/02/if-android-is-a-stolen-product-then-so-was-the-iphone/.

www.manaraa.com

out-of-the-box thinking, testing, and more testing.[62] Among the many projects that anticipated the iPhone's touch-sensitive screen were Microsoft's Surface, projects at MIT Media Lab and NYU's Interactive Technologies Program, restaurant service systems, package delivery-trackers, and various works existing somewhere in the space between art and function.[63]

Even if Apple was the first to sell a smartphone with the iPhone's precise feature set, Apple cannot plausibly claim that it was non-obvious to "combine references," *patentese* for applying inventions from distinct prior sources to a new field, or in new combinations.[64] Cellphones have become increasingly important in the United States and around the world; motivations to improve the operating system come from many places, including from the public demanding better phones and applications for them; hardware vendors wanting to sell phones; the carriers wanting to sell minutes and data-plans; application developers looking for strong platforms; and the prospect of complementary search, advertising, and increased consumer engagement.

Patent's defenders argue that patents serve a useful public disclosure function,[65] but what exactly do the touch patents disclose? Very little that is not already disclosed by the devices Apple sells on the public market. Play with an iPhone and you'll see how its interface works at the surface-level detail shown in the patent. Second-comers—whether imitators or independent discoverers of the utility of pressing virtual buttons—must perform their own experimentation, determine the proper parameters for the heuristics, squash bugs, and tame race conditions, all in the course of adapting the user interface to existing programs and hardware or building new ones around it. Showing these competitors a market opportunity does not pave the road there, either through the patent system or otherwise.

---

[62] *See* Buxton, *supra* note 61.

[63] *Id.*

[64] *See supra* note 30; *see also* Wyers v. Master Lock Co., 616 F.3d 1231, 1240 (Fed. Cir. 2010) (citing KSR Int'l Co. v. Teleflex, Inc., 550 U.S. 398, 402 (2007)) (describing the test for non-obviousness as a common sense inquiry into whether a person of ordinary skill in the art would have found it obvious to "combine references"; in order words, it would have been obvious to combine the disparate teachings concerning the subject matter to be patented in order to come up with the idea).

[65] *See infra* Section III.A.

www.manaraa.com

### *D. The Patent System Is Ill-Suited to Software*

The patent system is particularly ill-suited to software because the incentives it creates do not match the way software is developed. Instead of inducing optimal development of innovative software, the rules of the patent system encourage the production of patents. Where patents remain far from products—yet can be used to extract rents from those who make and sell products—it will frequently be entirely rational to set up in the business of "making patents" instead.

Taken to its extreme, we find the business of making patents in full force at Intellectual Ventures. That company, founded by former Microsoft CTO Nathan Myhrvold, is reported to host patent parties, inviting a group of smart people to sit around a table and generate patent-worthy ideas.[66] Sitting behind them, a group of lawyers transcribes these germs of ideas into patents. Rather than testing whether the ideas could be turned to practical use with staff or financing from Intellectual Ventures, the company makes license demands on others who have *already* found uses for the technology, unaided by the silly brainstorming of Intellectual Ventures. It is not the ideas, but their often-challenging development and implementation, that need incentive.

Imagine an incentive contract for a corporate manager that paid a bonus based on the number of *ideas* the manager had for product improvement, not on any measure of their implementation success. If corporate audits measured and rewarded the number of moleskine notebooks the manager filled with jottings rather than measuring the business division's output or profit, we could expect the manager to fill bookshelves and file drawers without limit. The firm might get some inkblots or brilliant doodles, elegant poetry, or doggerel, but it would be unlikely, except by coincidence, to induce much in the way of new corporate efficiencies or profits. The board of directors would be shirking its fiduciary duties if it failed to restructure its contracts to bring incentives closer to the firm's goals. Yet that is essentially the operation of our current patent system. It tells participants that by producing patents, they

---

[66] *See Inside Nathan Myhrvold's Mysterious New Idea Machine*, BLOOMBERG BUSINESSWEEK (July 2, 2006), http://www.businessweek.com/stories/2006-07-02/inside-nathan-myhrvolds-mysterious-new-idea-machine.

www.manaraa.com

can make money fast with the "right to exclude,"[67] so long as *someone* is producing products. In sum, Congress, as public fiduciary, is failing to "promote the [p]rogress of [s]cience and useful [a]rts" through the patent system.[68]

Patent incentives fail most dramatically in software because of the nature of software development, the low bar to patenting software features, and the gap in time and cost between initial software idea and its implementation in software design and programming.[69] Software and business method patents are problematic because they protect too early in the continuum from invention to innovation.[70] To obtain a patent, one need not have built a working model, nor have showed others how to get through the debugging stages; one can instead stake a claim on something little more than an abstraction. From there, the patent-holder might develop and commercialize a product, but he might also fail or take a detour, finding it more lucrative to look for that patented germ of an idea among others' products, even if those others had developed it independently.

As is, the patent system provides incentives where they are not needed: to generate yet more "ideas" without sufficiently encouraging the development of those ideas into useful products or services. Patent enforcement acts as an alternative to product development, rather than an inducement toward it.

## II. WHAT IS SOFTWARE?

### *A. Software Is Everywhere*

Software is the instructions given to a computing device to make it function.[71] Software is everywhere. In stand-alone

---

[67] "It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" Phillips v. AWH Corp., 415 F.3d 1303, 1312 (Fed. Cir. 2005) (quoting Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc., 381 F.3d 1111, 1115 (Fed. Cir. 2004)).

[68] U.S. CONST. art. I, § 8, cl. 8.

[69] *See infra* Parts II.A & II.B.

[70] *See infra* Part II.E (discussing software patent problems). With respect to business method patents, in *Bilski v. Kappos*, the Supreme Court affirmed the rejection of a patent on a business method, commenting that "some business method patents raise special problems in terms of vagueness and suspect validity." 130 S. Ct. 3218, 3229 (2010); 35 U.S.C.A. § 101 (West 2012).

[71] The Institute of Electrical and Electronics Engineers (IEEE) defines software as the "computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system." IEEE, STANDARD GLOSSARY OF SOFTWARE ENGINEERING TERMINOLOGY, (STD. 610.12-1990), at 66 (Sept. 28, 1990), *available at* http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-

www.manaraa.com

packages for home computers and business automation; embedded in devices large and small; customized for industrial uses; hosted "as a service," in the cloud.[72] Software simultaneously comprises infrastructure, the application-layer, and digital content; it is both product and component. It is ubiquitous and nearly universal yet highly specialized.

Software ranges from the simple (the command print("Hello World!") creates a functional software program in many coding languages)[73] to the complex (Microsoft Windows XP had a reported thirty-five million lines of code, while Vista contained more than fifty million).[74] While it might seem that Angry Birds and System/360 have as much in common as white mice and blue whales, there nonetheless exists a meaningful commonality between them. Just as both of those carboniferous life-forms share high-level traits based in evolutionary heritage and the pressures of natural selection, software on many scales shares common evolutionary features. Operating systems and drivers, programming tools (including compilers and debuggers), communications tools, and applications are all made from the same basic components. These components can be layered, so that an application can serve as a platform for plug-ins, which act as its own applications.[75] Software can be written at varying levels of abstraction, ranging from assembler instructions, human-readable source-code, to modeling languages that

---

glossary-610.12-1990.pdf. Within the software could lie any number of programming languages, which are composed of code setting out instructions for the operation of the computer. At least one statistical study of patents defines "software patents" by a set of keywords and categories, as the PTO does not classify "software" patents. This study intends to be inclusive, looking at patents read to bear on software, including those on software-using devices and methods generally implemented by software—those things it is possible to infringe by distributing or using software. *See generally* Sebastian von Engelhardt, *The Economic Properties of Software*, (Jena Econ. Research Papers, Working Paper No. 2008-045, 2008), *available at* http://hdl.handle.net/10419/25729.

[72] Martin Campbell-Kelly divides software among customization, OS, and packaged. MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 20 (2003).

[73] *See* The Linux Documentation Project, *The Linux Kernel Module Programming Guide: Modules vs Programs, § 3.1.2*, http://www.tldp.org/LDP/lkmpg/2.4/html/x437.html (last visited Feb. 25, 2013).

[74] *See* Steve Lohr & John Markoff, *Windows Is So Slow, but Why?*, N.Y. TIMES, Mar. 27, 2006, at C4, *available at* https://www.nytimes.com/2006/03/27/technology/27soft.html.

[75] While typically placed at the "application" layer of the OSI stack, software systems themselves are often layered, with operating systems sitting beneath applications, and "middleware." *See* United States v. Microsoft Corp., 253 F.3d 34, 53 (D.C. Cir. 2001) (discussing the distinctions among the various layers); *see also infra* note 274 and accompanying text.

www.manaraa.com

generate software from graphical input.[76] A proliferation of computing devices, driven down in cost according to Moore's law, has led to a vast increase in the development and use of software.[77] The net effect has been to produce miniaturization and embedded systems, on the one end, and "cloud" systems, on the other; all of which have extended software's scope.

Software is an information good. Like other information goods, most of its costs arise from fixed costs of development. The incremental costs of copying and distribution, however, are minimal.[78] Copyright addresses this type of good by prohibiting unauthorized copying of the final product. Yet while the direct copying of a computer program is cheap or free, there may be substantial additional costs to adapting software for use with new hardware, processes, and users. These challenges provide a window of first-mover advantage. Thus, even assuming that some of Google's smartphone features were inspired by Apple's iPhone, the fact that it took Google more than a year to move the first Android phone into production[79] demonstrates the unavoidable lag time in adapting hardware for new uses.

Adaptation also presents opportunities for complementary services, given that consulting and customization work can be done most effectively by those most familiar with the underlying software.[80] Adaptation knowledge is contextual, however, and is not encapsulated in patent documents, which conversely aim to "claim" the invention in the least instructive, generic terms.[81]

---

[76] *See* Keith Short, *Modeling Languages for Distributed Applications*, MICROSOFT DEVELOPER NETWORK (Oct. 2003), http://msdn.microsoft.com/en-us/library/aa302173.aspx.

[77] *See infra* note 240.

[78] The cost of exclusionary measures raises distribution costs. A producer restricting access must establish or pay for separate distribution infrastructure: access-controlled download sites, verification portals, or physical delivery with indicia of authenticity. *See, e.g.*, *How to Protect Software*, DIGITAL RIVER DEVELOPER RESOURCE, http://www.developer-resource.com/how-to-protect-software.htm(last visited Oct. 11, 2012). A producer distributing free software, however, can post it to third-party platforms such as sourceforge or github, which host open source code for free, or allow users to share the cost of distribution through peer-to-peer filesharing such as BitTorrent. *See infra* note 177. End-users are more likely to share the word-of-mouth marketing for a free product when they can link it directly. *See* Andrew Whitmore et al., *Open Source Software: The Role of Marketing in the Diffusion of Innovation*, 38 INFO. TECH. & CONTROL, no. 2 (2009).

[79] *See supra* note 40.

[80] *See* Josh Lerner & Jean Tirole, *Simple Economics of Open Source* 18 (HBS Finance Working Paper No. 00-059, October 2000), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=224008.

[81] Michael Risch, *The Failure of Public Notice in Patent Prosecution*, 21 HARV. J.L. & TECH. 179, 180 (2007) ("Patent applicants have an incentive to keep issued patent claims vague because vagueness allows for ex post gaming.").

www.manaraa.com

### *B. Who Makes Software?*

Our first impression of software development may focus on the firms who make and sell packaged software, such as Microsoft and Adobe. Yet much more software is produced by others: search engine and social media companies such as Google and Facebook write it to run their services; hardware manufacturers such as Apple write it to power their devices; non-software firms write (financial services), buy (law firms), or commission it to manage their internal operations. Businesses in most sectors of the economy now depend on software.[82]

Software-users make a lot of software. While very few individuals can (yet) make their own medicines successfully, many people and businesses are "user-innovators" of software, producing software to use themselves as a tool, as an intermediate step in some other process or function, or as a component, rather than to sell as a stand-alone product.[83] User innovators get "use value" from their innovations—a more direct source of value than the profit from sale to others—and they also benefit from closer access to "sticky information," i.e. information that is costly to acquire, transfer, and use.[84] As we will see, user innovators thus tend to approach intellectual property more openly, freely revealing their innovations and improvements.

Carliss Baldwin and Eric von Hippel show that lower communication costs—to which the Internet is key—bring open-collaborative innovation into competition with producer-driven innovation, while lower costs in software design technology make single-user innovators competitive.[85] Patents affect these classes of innovators differently. Whereas producer-innovators view patents as a means to secure profits,

---

[82] *See* Marc Andreessen, *Why Software is Eating the World*, WALL ST. J. (Aug. 20, 2011), http://online.wsj.com/article/SB10001424053111903480904576512250915629460.html.

[83] *See* Carliss Y. Baldwin & Eric von Hippel, *Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation*, 3-4 (Harvard Bus. Sch. Fin. Working Paper No. 10-038, 2009), *available at* http://ssrn.com/abstract=1502864. Eric S. Raymond says 95 percent of software is developed for in-house use—for its use value rather than for sale-value. Eric S. Raymond, *The Magic Cauldron* ch. 3, CATB.ORG, http://www.catb.org/esr/writings/magic-cauldron/magic-cauldron.html#toc1 (June 1999).

[84] *See generally* Joachim Henkel & Eric von Hippel, *Welfare Implications of User Innovation*, 30 J. TECH. TRANSFER 73 (2004); *see also* Eric von Hippel, *"Sticky Information" and the Locus of Problem Solving: Implications for Innovation*, 40 MGMT. SCI. 4, 429-39 (Apr. 1994).

[85] *See* Baldwin & von Hippel, *supra* note 83, at 2 ("[I]nnovation by individual users and also open collaborative innovation are modes of innovating that increasingly compete with and may displace producer innovation in many parts of the economy."); *see also id.* at 19-20.

www.manaraa.com

user-innovators are motivated by their own direct needs, even when those needs serve as intermediate steps or tools for producing something else for profit[86]—"scratching a developer's personal itch" as Eric Raymond puts it.[87] User-innovators have more efficient access to the "sticky information" about these needs and reap the entire benefit of meeting them.[88] Open collaboration is hindered, rather than helped, by patents, as innovators trying to share across firm boundaries are stopped by the firms' IP counsel[89] or subjected to burdensome NDAs and IP clearing processes.[90] As the Internet improves other conditions for open collaboration and user innovation, patent becomes a more salient hurdle.

### *C. How Is Software Made?*

Anyone starting out to develop software will typically have on hand its raw materials: a target computer system and a programming environment.[91] The developer will generate a problem statement, functional specification, or design requirements.[92] In some cases, this design will be implicit,

---

[86] *Id.* at 3-4.

[87] *See* ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY 23 (rev. ed. 2001).

[88] *See* William W. Fisher III, *The Implications for Law of User Innovation*, 94 MINN. L. REV. 1417, 1456-57 (2010). User-based incentives are more direct than the price signals of a market, since a user's knowledge of his demand need not be intermediated by the market's transaction costs. Prices, rightly appreciated by free-market economists for information exchange, are still a second-best to information derived from direct experience. *Cf.* Amy Kapczynski, *The Cost of Price: Why and How to Get Beyond Intellectual Property Internalism*, 59 UCLA L. REV. 970 (2012) (arguing that pricing itself adds cost and inefficiency to interaction).

[89] Andrew T. Pham & Matthew B. Weinstein, *Living with Open Source: Implementing, Managing and Enforcing A Uniform Policy for Your Enterprise*, 44 LES NOUVELLES 162 (2009), *available at* http://www.lesi.org/les-nouvelles/les-nouvelles-online/2009/september-2009/2011/08/04/living-with-open-source-implementing-managing-and-enforcing-a-uniform-policy-for-your-enterprise ("Because of the complexity and risks associated with open source—where source code is made freely available for all to review, edit, and use—many closed-source commercial enterprises discourage or prohibit use of open source; a common and short-sighted practice.").

[90] *See, e.g.*, Barbara LaSusa & Lawrence LaSusa, *Intellectual Property "Best Practice" Tips for Small Law Departments*, MICH. B.J., Jan. 2007, at 32; *see also* HENRY W. CHESBROUGH, OPEN INNOVATION: THE NEW IMPERATIVE FOR CREATING AND PROFITING FROM TECHNOLOGY 174-75 (2003).

[91] Many powerful programming languages and environments are available open source and free of charge. Web applications are often built on the wholly-free "LAMP" stack: Linux operating system, Apache webserver, MySQL database, and Perl or PHP scripting. *See generally* RAJ KAMAL, EMBEDDED SYSTEMS: ARCHITECTURE, PROGRAMMING, AND DESIGN (2d ed. 2008).

[92] *See* IIEE Computer Soc., *Guide to the Software Engineering Body of Knowledge*, §§ 1.1-1.6, 2, COMPUTER.ORG (2004), http://www.computer.org/portal/web/

www.manaraa.com

particularly for smaller individual projects. For longer-term projects and those requiring coordination among multiple programmers, more will be written down.[93] In either case, the original specifications often change as development progresses: new constraints appear or arise from implementation challenges; user demands change or expand; and available resources or hardware change.

Software developers treat as commonplace the proposition that the *idea* is only a small part of the final product. Ideas are a dime a dozen; everybody has them.[94] It is the implementation that remains difficult and time-consuming.[95] Between software ideas and successful deployment come many less glamorous but necessary steps: design, implementation, debugging, and testing (often in multiple rounds).[96] Even deployment may represent only a first step toward market acceptance; many software products launch with "beta" versions, and developers continue to tweak the product based on user feedback.[97] Web-based deployment, app stores, and

---

swebok/htmlformat. The IIEE's *Guide to the Software Engineering Body of Knowledge* was produced in an effort to systematize the discipline of software engineering. *Id.* at Introduction.

[93] *See, e.g.*, Alan M. Davis et al., *A Strategy for Comparing Alternative Software Development Life Cycle Models*, 14 IEEE TRANSACTIONS ON SOFTWARE ENG'G 1453 (1988); Zulkefli Mansor et al., *Towards the Development of Success Determinants Charter for Agile Development Methodology*, INT'L J. INFO. TECH. & ENG'G, at 1 (2011), *available at* http://www.academia.edu/662663/Towards_the_Development_of_Success_Determinants_Charter_for_Agile_Development_Methodology (contrasting agile software development methodologies requiring less planning up front with traditional "waterfall" development requiring phase-by-phase planning).

[94] So are books on entrepreneurship using that phrase. *See, e.g.*, Google search for "software ideas 'dime a dozen,'" GOOGLE, https://encrypted.google.com/search?tbs=bks%3A1&tbo=1&q=software+ideas+%22dime+a+dozen%22&btnG=Search+Books (search Google with the phrase "software ideas 'dime a dozen'").

[95] The website Halfbakery provides a great illustration, providing a "communal database of original, fictitious inventions, edited by its users." *More About the Halfbakery*, http://www.halfbakery.com/editorial/about.html (last visited Feb. 25, 2013); *see also* Clay Shirky, *LazyWeb and RSS: Given Enough Eyeballs, Are Features Shallow Too?*, OPENP2P.COM (Jan. 7, 2003), http://openp2p.com/pub/a/p2p/2003/01/07/lazyweb.html (describing a variety of web sites that permit people to submit ideas for software they think should exist and developers take the ideas and code them).

[96] *See* Davis et al., *supra* note 93, at 1453 fig.1.

[97] *See, e.g.*, *Customer Information Control System (CICS): Beta and Early Test Programs*, IBM.COM, http://www-01.ibm.com/software/htp/cics/betas.html (describing goals of beta release programs for CICS Products). The "beta" label may serve to signal the developers' commitment to continue developing the product, suggesting to users both that their feedback is welcome and that they should be more forgiving of rough edges. *See* Tim O'Reilly, *Design Patterns and Business Models for the Next Generation of Software*, O'REILLY.COM (Sept. 30, 2005), http://oreilly.com/lpt/a/6228. Some products, such as Google's Gmail, used this label for so long that commentators referred to them as "perpetual beta." Juliet Lapidos, *Why Is Gmail Still in Beta?*, SLATE.COM (Apr. 7, 2009, 6:33 PM), http://www.slate.com/articles/news_and_politics/explainer/2009/04/why_is_gmail_still_in_beta.html. Non-traditional software development

www.manaraa.com

networked computers and devices make it even easier for software producers to offer incremental in-place updates.

Narratives of software engineering and design describe the challenges of working on ideas "only slightly removed from pure thought-stuff."[98] Frederick P. Brooks's classic, *The Mythical Man-Month*, drew precepts from his experience managing the development of IBM's System/360 Operating System, a major project that was chronically behind schedule and over budget.[99] Among the many lessons Brooks presents, his instruction to "plan to throw one away" reminds us that prototypes are essential to good software design.[100] The first version shows the possibility (or futility) of a design; the second makes it practical. Programmers who get too attached to the first version may never get beyond it to attain a reliably functioning second iteration.

A quarter-century after Brooks's chronicle, Scott Rosenberg found that similar challenges beset the developers of the Chandler personal information manager.[101] Chandler was going to revolutionize calendar management and scheduling, then a tangle of independent applications still migrating between paper and computer.[102] Its developers proposed dozens of creative program features and inventive new designs for information management, drew up hundreds of project plans, and wrote thousands of lines of code.[103] Despite substantial work over three-plus years, most of the Chandler team disbanded without a mainstream product release, never crossing the gap from idea to successful implementation.[104] Some of its ideas, though, can be seen in both Apple's iCal and

---

paradigms—including agile development methodology; release early, release often; and open source—all recommend this. *See, e.g.*, Davis et al., *supra* note 93; Tim O'Reilly, *Design Patterns and Business Models for the Next Generation of Software*, O'REILLY.COM (Sept. 30, 2005).

[98] *See* FREDERICK P. BROOKS, THE MYTHICAL MAN-MONTH: ESSAYS ON SOFTWARE ENGINEERING 7 (1995).

[99] *Id.* at xiii. While the effort ultimately produced an operating system Brooks called "quite reliable, reasonably efficient, and very versatile. . . . The effort cannot be called wholly successful, however. . . . [T]he product was late, it took more memory than planned, the costs were several times the estimate, and it did not perform very well until several releases after the first."

[100] *Id.* at 116.

[101] *See generally* SCOTT ROSENBERG, DREAMING IN CODE (2007).

[102] *Id.* at 106-07; *see also* "JaredRhine," *Chandler Project Vision*, CHANDLER PROJECT (Sept. 10, 2010, 3:42 PM), http://chandlerproject.org/vision.

[103] ROSENBERG, *supra* note 101, at 20.

[104] The website where the open source software was distributed shows the last software update in April 2009. *Download Chandler Desktop 1.0.3*, CHANDLER PROJECT (July 26, 2010), http://chandlerproject.org/Projects/DownloadChandlerDesktop.

www.manaraa.com

Google Calendar. Had Chandler chosen to patent rather than follow an open route, its unsuccessful application development could still block or hold-up Apple's and Google's operational projects, both widely used today.

These examples, from very different software projects, illustrate the more general challenge of taking software from idea to implementation. It is never a simple matter of running "compile"; even the simplest programming project often requires multiple rounds of redesign, implementation, and debugging.[105] In Brooks's software development methodology, he allots just 1/6 of the time of building a software program to coding, with 1/3 for planning and fully half for testing, first of components and then of the entire system.[106] Following Google's lead, many software services now launch with "beta" tags[107] while learning from public testing.[108] Thus, a patent application

---

[105] *See generally* Davis et al., *supra* note 93. Because of the nature of software programming, in which there exists a large number of interrelated tasks, the probability that delays will occur is incredibly high. For instance,

> [i]n a single task, the assumption that all will go well has a probabilistic effect on the schedule. It might indeed go as planned, for there is a probability distribution for the delay that will be encountered, and 'no delay' has a finite probability. A large programming effort, however, consists of many tasks, some chained end-to-end. The probability that each will go well becomes vanishingly small.

BROOKS, *supra* note 98, at 15-16.

[106] *Id.* at 20.

[107] The Greek letters alpha and beta are used as indicators of product test phase. The alpha test is the prototype or internal "dogfood" version, while the beta test is often the first version released to outside testers. *See* Erik van Veenendaal, ed., *Standard Glossary of Terms Used in Software Testing*, INT'L SOFTWARE TESTING QUALIFICATIONS BD. (Oct. 19, 2012), http://www.astqb.org/documents/ISTQB_glossary_of_testing_terms_2.2.pdf. After Google, many companies have mixed some advertising into the labels, making invited beta testers feel like part of an exclusive club. *See* Lapidos, *supra* note 97.

[108] Software services, which can be updated on the server rather than client-side, can update their features in-use, with minimal impact on end-users. *See Windows Server Update Services Overview*, MICROSOFT TECHNET (Feb. 29, 2012), http://technet.microsoft.com/en-us/library/hh852345.aspx. Yet even downloadable software is moving toward more frequent incremental releases. *See, e.g.*, *Software Update*, APPLE, http://www.apple.com/softwareupdate/ (last visited Oct. 4, 2012) ("In addition to releasing new versions of the system software at regular intervals, Apple also releases a stream of free software updates to enrich your computing experience. Mac OS X automatically checks weekly for software updates provided you have an internet connection."); *Why Are There So Many Updates for Windows 7?: Reply to Posting of "AmandaPage,"* MICROSOFT COMMUNITY (Mar. 25, 2012), http://answers.microsoft.com/en-us/windows/forum/windows_7-windows_update/why-are-there-so-many-updates-for-windows-7/bd2bff08-0f45-451e-ac13-84cdaec4aa6d?msgId=0b83c4e8-4e38-4c51-a1bd-e76fa9821450 ("I'd like to believe the World's a good place but it's not. You've been hearing about 'hackers,' 'exploits,' etc. Many of these updates are to thwart these attacks on the system."). The Internet both facilitates and requires this updating: end-users have fast connections that make downloading updates easy, while network-exploitable flaws increase their urgency.

www.manaraa.com

filed sometime within the coding phase will reflect less than half the work required to achieve working implementation.[109]

In an effort to speed up the cycles from design to implementation, software developers respond with new methodologies, such as "agile development" and "extreme programming."[110] Modern software development methodologies build on the recognition that the conclusion cannot be seen from the start, even though goals may be fixed.[111] Instead of a single "waterfall," these methods recommend planning in pieces and break program development into sub-tasks that can be iterated individually.[112] The overall whole might be patentable, but each of those pieces is far too small to deserve a patent—yet that's the stage at which patents are claimed.[113] Patent incentives encourage inventors to *fix* more of the pieces at an early stage, thereby freezing the development before refactoring.

As another alternative to traditional software development models, many advocate modularity, similarly decomposing the program's function into chunks that can be upgraded without a wholesale rewrite.[114] Modular systems, separable along clearly defined interfaces, are preferred to the contrasting "spaghetti code," from which it is difficult to draw one strand without pulling up a tangle.[115] Modules may be as small as single functions,[116] making their reuse easy and more efficient than rewriting.[117]

---

[109] *See generally* Davis et al., *supra* note 93.

[110] *See* Mansor, *supra* note 93, at 1; VICTOR SZALVAY, AN INTRODUCTION TO AGILE SOFTWARE DEVELOPMENT (Nov. 2004), *available at* http://www.danube.com/docs/Intro_to_Agile.pdf.

[111] SZALVAY, *supra* note 110, at 7.

[112] Daniel B. Garrie, *The Legal Status of Software*, 23 J. MARSHALL J. COMPUTER & INFO. L. 711, 723 (2004).

[113] Kathryn Vesco Chelini, Comment, Pfaff v. Wells Electronics, Inc.*: Are You Risking Your Right to Patent Your Software?*, 37 NEW ENG. L. REV. 271, 297 (2003).

[114] Chris Falkowski, *Negotiating Intellectual Property Rights in Software-Related Contracts*, MICH. B.J., Aug. 2012, at 31, *available at* http://www.michbar.org/journal/pdf/pdf4article2071.pdf.

[115] When Netscape released its browser code to the public, in what would eventually become the Mozilla project, developers initially doubted they would be able to do anything with the code. *See* HENRIK INGO, OPEN LIFE: THE PHILOSOPHY OF OPEN SOURCE 102 (2006) ("After all the initial excitement, the Mozilla project proved, at least in part, to be a disappointment. It soon transpired that in the fight to beat Microsoft, Netscape's programmers had been pressed for time and had produced really confusing and low-quality code."); *see also* Andy Patrizio, *Netscape 6: Does Anyone Care?*, WIRED.COM (Apr. 5, 2000), http://www.wired.com/science/discoveries/news/2000/04/35350.

[116] *See supra* note 73.

[117] *See, e.g.*, *Modules and Packages for Code Reuse*, JYTHON PROJECT, http://www.jython.org/jythonbook/en/1.0/ModulesPackages.html#chapter-8-modules-and-packages-for-code-reuse ("Modules are helpful for creating libraries that can be imported and used in different applications that share some functionality.").

www.manaraa.com

### *D. Open Source and Free Software*

A growing body of software development comes from free and open source software communities.[118] Open source software is developed to be shared freely, with its source code attached to the program for others to modify.[119] Successful free and open source projects—including the Linux kernel and the operating systems around it, the Apache webserver, and numerous programming languages—do more than just share their source code; they build collaborative communities around code development.[120] Programmers contribute for a variety of reasons: personal satisfaction, reputation, and a desire to improve the functioning and reach of their own applications running on top.[121] By encouraging their employees to participate, companies contribute to or even lead open source efforts in order to find talent, enhance their own offerings, and improve the software that complements their products, whether those products involve enterprise hardware (IBM), mobile devices, or search advertising (Google).[122]

Open source works best when development is modular—when individual participants can communicate through patches and improvements. Projects learn to "release early and often."[123] Rather than saving up for the complete solution, they reward contributors and speed the test, debug, and improvement cycle, adding "more eyeballs" to the debug process.

Software development is cumulative—especially when modular—because it learns efficiently from what goes before and because it may, over time, incorporate improvements to

---

[118] While nothing requires that software developed for open source redistribution be made any differently from proprietary software, many of the most successful free and open source projects are community-driven. *See generally* Raymond, *supra* note 83. Many participants so associate this community-based mode with free and open source that they question whether Google's Android operating system is truly "open source," since while Google distributes the source code under an open license, it does not manage the code base in a community, nor readily accept patches and feature requests from outside of Google. *See* Scott Gilbertson, *Is Android Open?*, WIRED (Oct. 22, 2010, 6:10 PM), http://www.wired.com/epicenter/2010/10/is-android-open/.

[119] Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563, 597 (2004).

[120] *Id.* at 613.

[121] Karim R. Lakhani & Robert Wolf, *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*, *in* PERSPECTIVES ON FREE AND OPEN SOURCE SOFTWARE 11-12 (Joe Feller et al., eds., 2005).

[122] *See* Josh Lerner et al., *The Dynamics of Open Source Contributors*, 96 AM. ECON. REV. 114, 115 (2006)); Lakhani & Wolf, *supra* note 121, at 7, 11 (Rishab Ghosh on making money from OS software).

[123] *See* Raymond, *supra* note 87.

www.manaraa.com

individual components comprising the whole.[124] Merges and Nelson contrast cumulative inventions with discrete inventions, which tend to be stand-alone products and do not bear the same potential to "point the way to wide ranging subsequent technical advances" as do cumulative inventions.[125] Consistent with the benefits of the cumulative process, modularity lowers barriers to entry, as a newcomer only needs to create a component, and not an entire product from scratch. Many people have adapted David Weinberger's classic description of the Internet as "small pieces loosely joined" as an equally apt description of the modularity principle underlying the Unix philosophy.[126] Software progresses by sharing design patterns and anti-patterns[127] as well as modules of code. Yet these patterns remain abstract ideas, which are unfit for patent.

### *E. Where Does Patent Fit?*

The software industry is still young, and the form of software keeps changing as the machines for which it is written increase in power. Mainframes dominated the market only half a century ago, followed by the advent of desktops, and then portable devices pushed function back out. Now, it is not client-server so much as client-client-and-multiple-servers that predominates. Development changes with each of these movements, but the field changes in waves, not isolated sparks of genius. As new technologies or platforms appear, new programs and businesses become possible.[128] What was once unimaginable rapidly becomes obvious with the right tools. But the patent office, without the right search tools or industry

---

[124] Falkowski, *supra* note 114, at 31.

[125] *See* Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope*, 90 COLUM. L. REV. 839, 880 (1990).

[126] *See* Scott Merrill, *Everything Old Is New Again: Microsoft MinWin Attempts to Modularize Windows*, TECHCRUNCH (Nov. 18, 2009), http://techcrunch.com/2009/11/18/everything-old-is-new-again-microsoft-minwin-attempts-to-modularize-windows/; *see also* DAVID WEINBERGER, SMALL PIECES LOOSELY JOINED: A UNIFIED THEORY OF THE WEB (2002), *available at* http://www.smallpieces.com/index.php.

[127] The first wiki, long before Wikipedia, was MeatballWiki, designed to facilitate the sharing of software design patterns. *Meatball Wiki*, C2.COM, http://c2.com/cgi/wiki?MeatballWiki (last updated Sept. 20, 2012); *cf. generally* CHRISTOPHER ALEXANDER, A PATTERN LANGUAGE (1977) (proposing the notion of "pattern languages" in architecture, upon which software patterns were modeled); *see also* ERICH GAMMA ET AL., DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE, at foreword (1995).

[128] *See* Tim O'Reilly, *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, COMM. & STRATEGIES No. 65, at 17, 19 (2007).

www.manaraa.com

experience, sees novelty from the first person who brings in a new exemplar.[129]

As the Internet came into commercial use, tens of thousands of individuals were simultaneously experimenting with ways to use and build upon this network of networks. Some adapted old client-server architectures, others built new peer-to-peer designs. Whereas older patents may have disclosed "remote" access,[130] newer claimants would refer to "communications networks."[131] In either case, many were laying claim to combinations that were in fact made obvious by the new technology and, as a consequence of its obviousness, should have been held presumptively unpatentable.[132]

Software developers do not perform patent searches before developing products. This is a rational decision, especially in a post-*Seagate*[133] environment when the threat of treble damages for willful infringement looms slightly less large. Most rightly recognize they will not find useful information disclosed in patents, which are written for patent examiners (and to obscure any similarities to prior art). A search for patents would be less useful than reading algorithms and public code.

---

[129] Richard S. Gruner, *Better Living Through Software: Promoting Information Processing Advances Through Patent Incentives*, 74 ST. JOHN'S L. REV. 977, 1063-64 (2000) (focuses on lack of patent office abilities, more than threat of falsely thinking there is novelty—more directed at non-obviousness).

[130] E-Data asserted that its 1985 patent 4,528,643 "system for reproducing information in material objects at a point-of-sale location," claiming "information manufacturing machines located at point-of-sale locations and an information control machine located remotely," applied to online media purchases. U.S. Patent No. 4,528,643 (filed Jan. 10, 1983), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "4528643"); BESSEN & MEURER, *supra* note 13, at 8. It won numerous settlements before the patent was held not to stretch from retail-store kiosks to personal computer downloads. *Id.* at 9.

[131] *See, e.g.*, Amazon's "one-click" ordering patent, Method & System for Placing a Purchase Order Via a Communications Network, U.S. Patent No. 5,960,411 (filed Sept. 12, 1997) (issued Sept. 28, 1999), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "5960411"); MercExchange, LLC's Consignment Nodes, U.S. Patent No. 5,845,265 (filed Nov. 7, 1995) (issued Dec. 1, 1998), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "5845265") (claims include "[t]he apparatus of claim 1 wherein said communications means is via an internet").

[132] Patent law bases its obviousness judgments on the vantage point of a person "having ordinary skill in the art." KSR Int'l Co. v. Teleflex Inc., 550 U.S. 398, 399 (2007) (internal quotation marks omitted); *see also supra* note 30. The Internet connectivity and cheap computing revolution dramatically increased the number of such persons, and the numbers of those above average, so technological advancement increased the likelihood of simultaneous invention. *See* Mark A. Lemley, *The Myth of the Sole Inventor*, 110 MICH. L. REV. 709, 715 (2012).

[133] *In re* Seagate Tech., LLC, 497 F.3d 1360, 1371 (Fed. Cir. 2007) (holding that enhanced damages are available in patent infringement cases only upon a showing of objective recklessness).

www.manaraa.com

Yet, developers also face pressure to obtain software patents, especially at an extremely early stage of development. Despite the prohibition on "abstract idea" patents, most recently reiterated in *Bilski v. Kappos*,[134] many software patents describe abstract algorithms—useful algorithms, perhaps—but claimed at such a level of abstraction that both describes a fact of nature and requires substantial implementation to capture their utility. At this stage, the patented algorithm is neither useful in itself, nor does the patent serve as a useful incentive to the deployment or commercialization of useful software embodying it.

Software development is not just risky, it is fundamentally uncertain.[135] Even the most determined efforts have no definite probability of success. They depend on too many factors, including quality of programming, hardware fitness, market readiness, and consumer (network) whims.[136] The best chance of success in software resembles the approach to diversification in financial markets: try many options and see what works.[137] Software lends itself to this approach because its startup capital costs are low.[138] There is no grand accounting for all the failed projects—after all, most sourceforge offerings have few or no downloads[139]—which simply means that open source is not a magic bullet. By percentage and by the ratio of successes to costs incurred, open source does, however, come out well.[140]

Rapid prototyping, agile development, and "release early, release often" all share the ability to fail fast, leaving the

---

[134] 130 S. Ct. 3218 (2010). The *Bilski* Court recited from "precedent[] . . . the unpatentability of abstract ideas . . . ." *Id.* at 3229.

[135] *See generally* FRANK H. KNIGHT, RISK, UNCERTAINTY, AND PROFIT (1921).

[136] Many critics judged HP/Palm's WebOS superior to Apple's iOS; it offered multi-tasking sooner and an innovated user interface. *See* Pal, *supra* note 58. But HP partnered with a weak phone carrier, did not open application development soon enough, and has the division, leaving stacks of unsold phones and tablets. *See* Chen, *supra* note 31.

[137] *See* Gideon Parchomovsky & R. Polk Wagner, *Patent Portfolios*, 154 U. PA. L. REV. 1, 37-39 (2005).

[138] *See* George Anders, *How to Launch a Billion Dollar Startup on a Shoestring*, FORBES (May 2, 2012), http://www.forbes.com/sites/georgeanders/2012/05/02/thrifty-startup/.

[139] ROSENBERG, *supra* note 101, at 99 (Of the 100,000+ projects on the website, "[f]ew have a large population of users; a good number of them are simply one-person operations.").

[140] Martin Courtney, *Open Source vs. Proprietary Software*, COMPUTING.CO.UK (Apr. 5, 2011), http://www.computing.co.uk/ctg/feature/2040433/source-vs-proprietary-software (noting the lower total cost of ownership arguments in favor of adopting open source software).

www.manaraa.com

stage open for another round of experimentation.[141] Google's "20 percent time" reflects a similar move to take advantage of employees' latent knowledge and interests, allowing them to experiment while demanding results.[142] This does not work for developers who are stopped early on by patenting, especially if pre-failure patenting does not disclose real *utility* information to the market. If Chandler's developers had been patent-oriented, they would have applied for patents while drafting their design sketches.

## III. PATENT THEORY

In 1958, economist Fritz Machlup took on a Senate commission to perform "An Economic Review of the Patent System." He concluded that review with the equivocal statement,

> [i]f we did not have a patent system, it would be irresponsible, on the basis of our present knowledge of its economic consequences, to recommend instituting one. But since we have had a patent system for a long time, it would be irresponsible, on the basis of our present knowledge, to recommend abolishing it.[143]

In the half-century since then, scholars have taken up the call to develop empirical studies of the patent system to improve our "present knowledge," while also clarifying the theories by which patent might operate.

The Constitution authorizes Congress to issue patents—"[t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[144] Courts and academic analysts have advanced several theories

---

[141] By contrast, Microsoft's re-envisioning of the operating system, "HailStorm," was described as one large project. *See* Joe Wilcox, *Microsoft's HailStorm Unleashed*, CNET (Mar. 19, 2001, 12:25 PM), http://news.cnet.com/2100-1001-254337.html. Faced with a complex task and all the backwards-compatibility challenges of a second-generation product, the project was significantly delayed from predicted estimates, and scaled back greatly by the time of its release. *Id.*

[142] *See* LEVY, *supra* note 32, at 124; Google: Official Blog, *Google's "20 Percent Time" in Action*, GOOGLEBLOG (May 18, 2006), http://googleblog.blogspot.com/2006/05/googles-20-percent-time-in-action.html. Google is also obsessed with usage testing and data, running A/B tests on the shades of blue or number of links on a screen. *See* Charles Arthur, *Marissa Mayer's Appointment: What Does it Mean for Yahoo?*, GUARDIAN (July 16, 2012, 6:35 PM), http://www.guardian.co.uk/technology/2012/jul/16/marissa-mayer-appointment-mean-yahoo; Laura M. Holson, *Putting a Bolder Face on Google: In Silicon Valley an Executive Turns Celebrity*, N.Y. TIMES, Mar. 1, 2009, at B.1 & B.8.

[143] S. SUBCOMM., ECONOMIC REVIEW, *supra* note 1.

[144] U.S. CONST. art. I, § 8, cl. 8.

www.manaraa.com

to explain what promoting progress means in the patent context. Machlup identified the theories as the "monopoly-profit-incentive" theory, "natural law," "reward-by-monopoly," and "exchange-for-secrets."[145] William Fisher similarly identifies "utilitarian," "labor," "personality," and "social-planning" theories.[146] In recent scholarship, as well as courts and policy debates, the utilitarian incentive theory has taken primary focus.[147]

Theory, of course, can be either descriptive or normative. It can try to map the world of innovation as it exists or chart a course for future growth. In either case, it requires an explanatory component, the step in which the features of the theory are levered against their environment to produce impact.[148] For theories that developed to explain patents generally, we ask whether the same leverage points and mechanisms also apply to software inventions.

### *A. Incentive Economics*

If we take the incentive function of patents seriously, we need to ask not only whether the system works to promote innovation but also by what mechanism it does so. Incentives can work either directly or indirectly. If direct, we identify something we want and pay for it, rewarding its producers, who increase their productivity in order to earn greater rewards. The patent system works in the realm of indirect incentives, granting patents not because patents *per se* are inherently valuable, but to promote the "progress of useful arts," or innovation. Because we want something—innovation—we cannot measure or compensate directly, we reward patent-filers with idea-exclusivity as a proxy for

---

[145] *See* S. SUBCOMM., ECONOMIC REVIEW, *supra* note 1, at 21; *see id.* at 33 (concluding that "[t]he thesis that the patent system may produce effective profit incentives for inventive activity and thereby promote progress in the technical arts is widely accepted. This is regarded as the fundamental economic justification of patents.").

[146] *See* William Fisher, *Theories of Intellectual Property*, *in* NEW ESSAYS IN THE LEGAL AND POLITICAL THEORY OF PROPERTY 168, 173 (Stephen R. Munzer ed., 2001).

[147] *See* S. SUBCOMM., ECONOMIC REVIEW, *supra* note 1, at 33; *see also* A. Samuel Oddi, *Un-Unified Economic Theories of Patents—The Not-Quite-Holy Grail*, 71 NOTRE DAME L. REV. 267, 277-81 (1996) (discussing other theories and critiques of the "patent-induced theory"); scholarship cited *infra* notes 151-57.

[148] For fans of South Park, this is the missing "Phase 2" in the Underpants Gnomes' business model. Phase 1: Collect underpants, Phase 2: ?, Phase 3: Profit. *South Park: The Underpants Business* (Comedy Central television broadcast Dec. 16, 1998), *available at* http://www.southparkstudios.com/clips/151040/the-underpants-business; *Gnomes (South Park)*, WIKIPEDIA.ORG, http://en.wikipedia.org/wiki/Gnomes_%28South_Park%29 (last visited Sept. 7, 2012).

www.manaraa.com

innovative activity. With this layer of indirection, however, we lose some of the incentive's fidelity to its underlying purpose and thereby introduce inefficiencies and errors. In particular, as economists commonly measure the effectiveness of incentives in principal-agent relationships, we can analyze some of the effectiveness of the patent system as a matter of agency costs: how effectively is the principal—the American public—getting innovation from the agents-developers to whom it offers patent incentives, and does the rate of innovation sufficiently outweigh patent's costs? We gain insight into the choice by reviewing the agency costs.

As an economic matter, patent offers a tradeoff between dynamic benefits and static costs. By giving patent-holders a twenty-year right to exclude, the patent system exchanges static *ex post* costs, which include the deadweight loss of monopoly pricing and restricted opportunities for competitors to build upon a patented invention, for dynamic *ex ante* benefits, such as the encouragement of invention and its disclosure to the public.[149] Because information is a public good, inventions can be copied more cheaply than they can be developed. As a result, people responding to economic incentives would not invest "enough" in invention without some legally enforced right to exclude.[150] Patent's right to exclude provides inventors an opportunity to appropriate more of the returns from investment in research and development.[151]

The basic utilitarian theory, then, justifies patents as reward-based incentives for inventive activity. Inventors are motivated by patent's exclusive right to invest in research and development, to innovate, and to disclose to the public, more than they would under alternative regimes, namely copyright and trade-secret's non-disclosure. Without the appropriability afforded by patent, inventors would refrain from any investment that an imitator would not also have to make.[152] This presents a particularly acute concern in the case of pharmaceuticals, where it can cost millions of dollars to

---

[149] DAN L. BURK & MARK A. LEMLEY, THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT 71-72 (2009); JOHN B. TAYLOR & AKILA WEERAPANA, PRINCIPLES OF MICROECONOMICS 268 (7th ed. 2012).

[150] *See generally* TAYLOR & WEERAPANA, *supra* note 149.

[151] *Id.*

[152] *See generally* David J. Teece, *Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy*, 15 RES. POL'Y 285 (1986); Kenneth J. Arrow, *Economic Welfare and the Allocation of Resources for Invention*, *in* THE RATE AND DIRECTION OF INVENTIVE ACTIVITY 609-25 (1962).

www.manaraa.com

discover, test, and develop a drug[153] but only pennies to reproduce it.[154] Patent exclusivity (along with any statutory or FDA extension of the term) gives the inventor an opportunity to recoup investment by charging more than the minimal marginal cost of reproducing the invention.[155]

Under a pure reward theory, society should pay inventors (either through direct or indirect incentives, with the resulting foregone consumer surplus) only to the extent necessary to induce innovation.[156] Yet, many patents are granted with no proof that they were necessary but-for causes of the innovation, and patents often turn out to be useless, worth less to their owners (not to mention society as a whole) than they cost. Hence, scholars have developed increasingly complex mechanisms to explain away the patent paradox, why inventors seek patents, and why society grants them.[157]

Prospect Theory: Staking a Claim

Edmund Kitch's influential "prospect theory" responded to this mismatch by suggesting that patent's incentive functions not through the direct right to exclude but as a coordinating mechanism for related activity.[158] In Kitch's theory, the patent claimant, like the prospector who identifies and stakes out a potential mineral claim, earns the right to control all activity within that domain, whether or not he knew its extent at the time.[159] Thus, the patent properly claims even activity unanticipated at the time of the patent application while blocking others from using the initial invention.

According to Kitch, this prospecting induces the optimal level and division of activity because the patent-holder is put in a position to coordinate the research efforts of others. Although

---

[153] BURK & LEMLEY, *supra* note 149, at 39.

[154] *Id.* at 71.

[155] *Id.*

[156] *See* Nicholas Economides & William N. Hebert, *Patents and Antitrust: Application to Adjacent Markets*, 6 J. TELECOMM. & HIGH TECH. L. 455, 464 (2008) ("For the duration of the patent monopoly, society temporarily trades the loss in consumer surplus for the adequate incentive for company *A* to invent product *A*.").

[157] Professors Hall, Ziedonis, Wagner and Parchomovsky run through several of the accounts that reach the conclusion of "paradox." *See* Bronwyn H. Hall & Rosemarie Ham Ziedonis, *The Patent Paradox Revisited: An Empirical Study of Patenting in the U.S. Semiconductor Industry, 1979–1995*, 32 RAND J. OF ECON 101 (2001); Parchomovsky & Wagner, *supra* note 137.

[158] *See* Edmund W. Kitch, *The Nature and Function of the Patent System*, 20 J.L. & ECON. 265, 268 (1977).

[159] BURK & LEMLEY, *supra* note 149, at 69-70.

www.manaraa.com

development might be incomplete at the time of patent grant, he argues that early patenting increases efficiency.[160] Staking the claim leaves a marker pointing to its proprietor, putting others on notice of a prior interest. Whether by conducting research around his patent claims or licensing that activity to others, the inventor/patent-holder's coordination reduces duplicative research efforts.[161]

But Kitch's theory draws heavily on the Coasean counterfactual, in which transaction costs are low and information easily available.[162] On this premise, Kitch argues that the patent-holder "coordinate[s] the search for technological and market enhancement of the patent's value so that duplicative investments are not made and so that information is exchanged among the searchers."[163] Before the coordination function can arise, however, both the patent-holder, and, more importantly, the rest of the field's innovators must know where the patent's claims lie and whether their work approaches or encroaches. This demarcation function operates particularly poorly in the context of software patents.

Patents do not serve a property-rule's notice function well in many industries, as Bessen and Meurer have documented.[164] The failure is heightened in software, where the language of claims is far removed from that of research and development.[165] Bessen and Meurer argue that patents fail as property particularly because they fail to give adequate notice of their boundaries.[166] The fuzziness of claims leads to high clearance costs, failure of the public disclosure function, uncertainty for both first- and second-comers, and thickets. In Bessen and Meurer's study, these problems are particularly acute for software patents, where the art is both newer and more abstract. Even if software developers were to search the patent literature (against the advice of counsel, who generally

---

[160] *Id.* at 70.

[161] *Id.* at 276.

[162] *See* Dan L. Burk & Mark A. Lemley, *Policy Levers in Patent Law*, 89 VA. L. REV. 1575, 1602 (2003) ("The Coase theorem is doing Kitch's work here.").

[163] *See* Kitch, *supra* note 158, at 276.

[164] *Cf.* BESSEN & MEURER, *supra* note 13, at 43 (discussing "analogous problems that arise during the acquisition of new mineral rights").

[165] *See* Kirby Ferguson, *Everything is a Remix: Part 4*, YOUTUBE (Feb. 16, 2012), http://www.youtube.com/watch?v=yAmmtCJxJJY (describing "information manufacturing machine, which covers anything computer-like") (referring to the patent disputed in *Interactive Gift Express, Inc. v. CompuServe Inc.*, 256 F.3d 1323, 1338-39 (Fed. Cir. 2001); U.S. Patent No. 4,528,643 cols. 4-5 (issued July 9, 1985), *available at* http://patft.uspto.gov/netahtml/PTO/srchnum.htm (search "4528643").

[166] BESSEN & MEURER, *supra* note 13, at 53.

www.manaraa.com

caution that search just invites claims for willful infringement), the programmers would be unlikely to find the patents that could later be claimed against them.[167]

Kitch proposed that patents serve to coordinate activity, but more efficient coordination mechanisms exist both within and outside patent.[168] Patents can become more salient coordination tools when pooled, either as part of a single company's portfolio or an industry patent pool. Moreover, innovators may find that non-market coordination mechanisms, aided by new communications technologies, make patent-free coordination more effective.

### *B. Patent as Coordination*

Polk Wagner and Gideon Parchomovsky argue for a "portfolio theory" of patents.[169] A company can be justified in acquiring patents that are individually worthless, they say, if those patents fit "into a collection of related patents."[170] "The heft of a patent portfolio can provide a firm with a strong market position (either real or perceived) in a particular field, thus encouraging upstart innovators to combine their inventions with that of a portfolio holder, rather than seeking to develop their own market niche."[171] Like a financial portfolio, the patent holdings gain value and mitigate risk through diversity.[172] The portfolio may send a stronger signal to would-be competitors than an individual patent would, especially when combined with an effort by the firm to advertise its portfolio strength.[173] These "super-patents," advocates point out, have distributional effects.[174] Large companies are better able to

---

[167] Nor does the patent-coordinated development model even make sense for software. Burk and Lemley say it applies most convincingly to pharmaceuticals, where drugs can be accurately and concisely described. *See* BURK & LEMLEY, *supra* note 149, at 26-27. There, a patent covers the molecule for a particular use. *See id.* Others contemplating the same or similar uses would have to engage in similar exploration and testing, much of which they can save by licensing the patent. In software, even adapting software covered by a patent to the same use made by its inventor (which may not be the use described in the patent disclosure) requires its own implementation and testing. *See infra* Part IV.C.1, discussing first-mover advantage and reverse engineering. There's much to be gained from software reuse, but that is distinct from patent reuse.

[168] *See generally* Kitch, *supra* note 158; *see also infra* Part III.B-C.

[169] *See generally* Parchomovsky & Wagner, *supra* note 137.

[170] *Id.* at 5-6.

[171] *Id.* at 33-34.

[172] *Id.* at 37-38.

[173] Various firms are said to walk into meetings with stacks of patents, saying in effect "we'll find something to hurt you with in here."

[174] *Id.*

www.manaraa.com

acquire sizeable portfolios, while smaller companies and new entrants may patent strategically to aim at gaps.[175]

On the flip side, patent-coordination imposes a sharp opportunity cost because patent's advantage as a coordination mechanism now, and in software-related fields, drops in the face of new coordination alternatives. The Internet and electronic communications technologies decrease the costs of non-patent coordination, enhancing the role of what Yochai Benkler terms "social production."[176] As the general transaction costs of communication and creation fall, inventors can engage in lightweight collaboration more easily. Instead of the organizing force of a firm, with contracts, NDAs, and overhead, they may choose to self-organize through mailing lists and wikis. Software developers share information, just not through patents; they post and respond to inquiries on StackExchange, github, wikis, mailing lists, forums, blogs, and microblogs, most discoverable with a web search such as Google or Bing.[177] To this non-market coordination, patent comes as a hindrance, not an aid.

Patent defense is costly,[178] avoidance impossible, and so the transaction costs of property rights mount even for those who wish to work outside the patent system. For example, it takes several contacts, license agreements, and royalty payments to join MPEG-LA, the licensor of the AVC patent pool around the H.264 video codec.[179] It takes only a source code checkout to join Xiph.org in developing or using Theora video,[180] but MPEG-LA's CEO asserts that users of other codecs face latent patent

---

[175] *See id.* at 53-56.

[176] *See generally* YOCHAI BENKLER, THE WEALTH OF NETWORKS: HOW SOCIAL PRODUCTION TRANSFORMS MARKETS AND FREEDOM 91-128 (2006), *available at* http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf.

[177] See the many programming-language-specific instances of collaboration at *Programmers*, STACKEXCHANGE, http://programmers.stackexchange.com/ (last visited Feb. 26, 2013); GITHUB, https://github.com/ (last visited Feb. 26, 2013), SOURCEFORGE, http://sourceforge.net/ (last visited Feb. 26, 2013), and *Google Code*, http://code.google.com/ (last visited Feb. 26, 2013), are among many online source code repositories or searches. Starting a project at SourceForge and similar sites will instantiate a repository, wiki, forum, and mailing lists, any of which may be used for collaborative development. When these are available on the open web, as many are, programmers asking similar questions in entirely disparate fields may find them to share information.

[178] *See supra* note 13.

[179] *See* Tim Siglin, *The H.264 Licensing Labyrinth*, STREAMING MEDIA MAG., Feb./Mar. 2009, at 80, *available at* http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-H.264-Licensing-Labyrinth-65403.aspx.

[180] *See* THEORA, http://www.theora.org/ (last visited Aug. 24, 2012).

www.manaraa.com

conflicts.[181] The costs of negotiating licenses even among themselves may swamp the benefits of informal collaboration, while developers who agree among themselves to forego patent claims may still face patent threats from outside. Many, for example, talk about the problems of clearing rights from universities or employers in unrelated fields—potential claimants who may have no plans or opportunities to exploit the ideas but nonetheless fear "losing" them.[182] The costs of patent-property ownership outweigh its benefits in lightweight collaborations.

Patent thus provides suboptimal coordination. Moreover, it is insufficient glue around which to coordinate because a patent is not a right to manufacture, that can thus be "blocked" in its scope by both prior and subsequent inventions.

### *C. Patent Races*

John Duffy argues that even if prospect-based coordination fails, the system nonetheless serves economic efficiency by inducing races to patent early, dissipating the monopoly rents and producing the social benefit of earlier innovation.[183] In this model, the system correctly encourages patenting at an early stage, because the multi-party race to invent, and to claim that invention well before it is commercially viable, shortens the patent's lifespan of exclusivity and sends the invention into commercial use, and then public domain, faster than it would otherwise appear. "The prospect features of the patent system are important not for suppressing or avoiding rivalry, but for accentuating and directing it."[184] Duffy refines the model from earlier assumptions, that a course of research is known and merely

---

[181] Jan Ozer, *Ogg, MPEG LA, and Submarine Patents*, STREAMINGMEDIA.COM (Mar. 4, 2010), http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=65782 (interview with MPEG-LA CEO who expresses concern over latent patent conflicts, or "submarine patents").

[182] *See*, *e.g.*, Limor Fried, Open Hardware Summit 2010 Keynote Address: Why Do Open Source Hardware? (Sept. 23, 2010), http://www.adafruit.com/blog/2010/09/27/limor-ladyada-fried-open-source-hardware-summit-keynote-hd-video/. No one wants to be in the tech transfer office that signs off on free use of the next Gatorade.

[183] John F. Duffy, *Rethinking the Prospect Theory of Patents*, 71 U. CHI. L. REV. 439, 443 (2004). *But see* Michael Abramowicz, *The Uneasy Case for Patent Races Over Auctions*, 60 STAN. L. REV. 803, 811-13 (2007) (suggesting that patenting early can reduce incentives to innovate/commercialize at the end of the patent term); Michael Abramowicz, *The Danger of Underdeveloped Patent Prospects*, 92 CORNELL L. REV. 1065, 1065-67 (2007) (patents obtained for their option value may not be developed if later contingencies do not occur).

[184] Duffy, *supra* note 183, at 479-80.

www.manaraa.com

awaiting the inventor's time, to a probabilistic model.[185] To avoid wasteful duplication of effort, he stresses the importance of early dissemination, either through patent application or through other information channels.[186] Yet here, especially once we have given up on patent as good coordination, the expiration of one patent tends not to clear the patent thicket. The racers barely clear the gate before another crowd is upon them.

Focusing on commercialization, Ted Sichelman notes that Duffy's early patenting provides little incentive for the significant follow-up work required to take an invention from conception to market.[187] Rather than deferring the patent application process, he recommends decoupling the invention and commercialization phases of patent.[188] He would offer a second grant of protection for the "commercialization" phase.[189] While this addresses a hole in earlier discussion of patent timing, it does so by compounding the problems of loose patents, leaving the earlier non-commercialized patents in front of any deployment activity.[190] Perhaps a provisional patent application, effective only if completed with evidence of commercialization, could serve both funding and deployment functions.

Christopher Cotropia identifies related problems as "the folly of early filing."[191] In particular, he follows Martin and Parnoy in "analogiz[ing] patent[s] to real options" in the world of finance as a tool for analyzing the decision to either forgo the option by abandoning the patent or to exercise it through commercialization (or alternatively, enforcing the rights of the

---

[185] *Id.* at 480-83.

[186] *See id.* at 482-83.

[187] Ted Sichelman, *Commercializing Patents*, 62 STAN. L. REV. 341, 389 (2010).

[188] *Id.* at 400.

[189] *See generally id.*

[190] The problems of patent-induced uncertainty are further compounded in Sichelman's proposed "probabilistic patent races." Ted Sichelman, *Probabilistic Patent Races* (draft presented at IPSC 2010), *available at* http://www.law.berkeley.edu/files/bclt_IPSC2010_Sichelman.pdf (last visited Oct. 31, 2012). As with Lemley's "rational ignorance" of bad patents, Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495, 1517 (2001), the theory largely neglects the chilling effect that assertions of legal claim can have on activity. *Cf.* Wendy Seltzer, *Free Speech Unmoored in Copyright's Safe Harbor: Chilling Effects of the DMCA on the First Amendment*, 24 HARV. J.L. & TECH. 171, 176 (2010) (lawful activity may be chilled by even invalid legal threats).

[191] Christopher A. Cotropia, *The Folly of Early Filing in Patent Law*, 61 HASTINGS L.J. 65 (2009).

www.manaraa.com

non-commercialized patent through litigation).[192] Cotropia usefully distinguishes the serious problem of non-practicing entities, recommending elimination of the "constructive reduction to practice" requirement to delay filing until the inventor provided actual working code.[193] Yet, he does not go far enough to curb the excesses of *practicing* trolls who exceed patent's bounds of utility. Moreover, leaving statutory bars and increasing demands of patent, without increasing the minimum quantum of patented invention, could increase secrecy and decrease sharing.

### *D. Patent Signals*

Clarisa Long's *Patent Signals* suggests a very different operation, in which patents are valued not for their exclusionary grant, but as signals, particularly in markets where other information about business strength or innovative promise is lacking.[194] Startups seeking venture funding have a limited reputation on which to draw, as compared to publicly traded companies with a long history in the business, and so they may seek patents as a way of signaling their innovative qualities to potential investors.[195] This signal is costly (albeit less costly to innovative firms than to "boring" ones), both to the signaling companies and to society, if it causes firms to spend their money wastefully.[196] The signal is also societally

---

[192] *See id.* at 95 n.208, 108. Under this analogy, the costs of obtaining the patent represents the purchase price of the option, and the costs of commercialization represent the cost of exercising the option. *Id.*

[193] *Id.* at 120. Cotropia provides a helpful summary of the distinctions between conception and reduction to practice: "The process of invention involves two steps—conception and reduction to practice—and is not considered completed until the performance of the second step." *Id.* at 72. However, the second step does not require actual reduction to practice. *Id.* at 73. Instead, "constructive reduction to practice," which refers to the moment when the invention is described in the patent application in a way that satisfies the requirements of 35 U.S.C. § 112 of "describ[ing], enabl[ing], and convey[ing] the best mode of the invention." *Id.* By contrast, "actual reduction to practice" would include "physically implementing the invention . . . and demonstrating the invention works as intended." *Id.*

[194] *See generally* Clarisa Long, *Patent Signals*, 69 U. CHI. L. REV. 625, 637 (2002).

[195] *See id.* at 672 ("Informational asymmetries are particularly acute when the company is a start-up firm. Start-ups have short histories, no market reputation, and frequently are in niche markets, often in fast-moving industries . . . . All else equal, we would expect the marginal benefit of a credible means of conveying information to be higher for a firm that has no market reputation than for a firm that can rely on an established reputation.").

[196] *See id.* at 660 ("This signaling approach is individually rational but socially inefficient. If boring firms can falsely match the signal sent by innovative firms by obtaining patents, this imposes an externality on innovative firms. The inefficiency arises not just because boring firms spend resources falsely matching innovative firms'

www.manaraa.com

costly in creating patent thickets: the patent that was obtained predominantly for its signaling nature persists for its 20-year lifetime and, particularly if the company is unsuccessful (i.e. the signal was misleading), it can be purchased and weaponized by patent trolls.[197]

Some economists reject the economic justifications for patent entirely. Boldrin and Levine, in *Against Intellectual Monopoly*, argue that justifications based on high fixed costs and competitive pricing approaching marginal cost prove too much: so long as imitators bear fixed costs of imitation, the same barriers that might deter innovation should deter imitation.[198] Since we see both innovation and imitation, we should instead recognize that a lead-time advantage, by means of a shorter and more contingent exclusivity than patent, often provides adequate incentive.

What is the scarce resource that would be under-produced but for the patent incentive? Information about needs? Coordination? Generation of ideas? Commercialization? Dissemination? Or none of the above?

Each of these theoretical treatments focuses on a different perceived imperfection or scarcity in a property-less information market. Introducing patents improves efficiencies by coordinating, forcing information, or concentrating attention and effort in the place where it would be under-produced if its producers were unable to appropriate profits through exclusive rights. But there is no reason to believe that the scarcities and inefficiencies that existed in prior technological eras will match those of the present day, especially in the context of computer technology. It is to those specifics that we now turn.

Burk and Lemley argue that each of these theories and accounts of innovation describes a different underlying

---

signals but also because innovative firms may signal excessively in an attempt to distinguish themselves from boring firms.").

[197] *See* Steven Levy, *The Patent Problem*, WIRED (Nov. 13, 2012, 6:30 AM), http://www.wired.com/opinion/2012/11/ff-steven-levy-the-patent-problem/all/ ("Because [the trolls] don't create anything, they can't infringe on anyone else's patents, no matter how overblown. That means they can't be countersued. This isn't mutually assured destruction; it's asymmetric warfare."); *cf.* Jennifer Urban, *Nuclear Deterrence for Patents: Let's Create a Network of Defensive Patents*, WIRED (Feb. 21, 2013, 6:30 AM), http://www.wired.com/opinion/2013/02/nuclear-deterrence-for-patents-lets-create-a-network-of-defensive-patents/ ("By de-weaponizing patents this way, the [Defensive Patent License] could help limit lawsuit risk. This is especially true for the risks posed by patent trolls, because a patent that can only be used defensively against a large network of innovators is likely to have little value for a troll.").

[198] *See* MICHELE BOLDRIN & DAVID K. LEVINE, AGAINST INTELLECTUAL MONOPOLY 184-208 (2008).

www.manaraa.com

technology field.[199] They recommend judicial tailoring.[200] Provided that judges can both distinguish cases consistently in practice while signaling to the field that they are doing so in an effort to disfavor software patents, judicial tailoring could indeed serve as an effective means of changing the law.

## IV. PATENT PROBLEMS IN SOFTWARE

Some previous scholarship has considered the industry- or technology-specific differences in patent's theory and application, notably Mark Lemley, with Julie Cohen[201] and Dan Burk,[202] and James Bessen and Michael Meurer.[203] Michael Carroll assesses the uniformity costs of intellectual property law against the simplifying benefits of uniform rules.[204] Here, I take these explorations further with deeper investigation of the nature and practice of the specific field of software development, evidence from recent survey reports, and a synthesis of these insights with theoretical accounts.

Considering how patents are theorized to work, we must examine these theories against the software practice, asking whether any of these mechanisms produces incentives that are appropriate for the development of software and the industries that use it. Using software patents to create incentives might fail in two directions: by encouraging the production of the wrong thing—namely, the production of patents—or by failing to encourage useful innovation through increasing impediments to it instead. Happily, the news is not all bleak. We do see innovative software. Many startups launch with new software products, and established companies still develop new and existing software products, both with and without patents on them. Nevertheless, we also see evidence that by encouraging the creation and use of patents, the system often interferes with innovation.

---

[199] *See* Burk & Lemley, *supra* note 162, at 1615.

[200] *Id.* at 1638-39, 1675; *see also* BURK & LEMLEY, *supra* note 149, at 95, 108-09, 167-70.

[201] *See* Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1 (2001).

[202] *See* Dan L. Burk & Mark A. Lemley, *Is Patent Law Technology-Specific?*, 17 BERKELEY TECH. L.J. 1155 (2002).

[203] *See generally* BESSEN & MEURER, *supra* note 13, at 187-214.

[204] *See* Michael W. Carroll, *One for All: The Problem of Uniformity Cost in Intellectual Property Law*, 55 AM. U. L. REV. 845 (2006).

www.manaraa.com

Reports from the Berkeley Patent Survey[205] confirm some intuitions about software patents' failures while challenging other assumptions. The survey finds that a relatively large percentage of entrepreneurs in the software field obtain patents,[206] contrary to the expectations of some critics. Yet, at the same time, these entrepreneurs "report that patents offer relatively mixed to weak incentives to engage in innovation."[207] Many of the same entrepreneurs view patents as valuable for obtaining funding, either from venture capitalists or friends and family.[208]

This funding-value is a circular way of measuring patent's social value, since entrepreneurs' perceptions of investors' wants may reflect neither the investors' actual criteria nor the underlying benefit of the patent.[209] Nonetheless, it is consistent with Long's theory of patent signals.[210] Indeed, the less sophisticated investors, like the friends and family of the developer, might be *particularly* prone to rely on heuristic signals of quality.[211] Unprepared to assess the underlying financial modeling or business strategy of a friend or relative's business, unsophisticated investors might instead view patents as indicators of the seriousness of the enterprise. Increasingly, however, venture capitalists in software-related fields are questioning the link between patents and value.[212]

### *A. Second-Order Incentives*

Taking the patent system's incentive function seriously requires analyzing not just whether but *how* incentives work, and to what end. Innovation creates social benefit through the

---

[205] *See generally* Stuart J.H. Graham et al., *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L.J. 1255 (2009).

[206] *Id.* at 1277-79 (indicating that 24% of software/Internet firms from the general population, and 67% of venture-backed software/Internet firms reported holding patents or patent applications).

[207] *Id.* at 1283. Specifically, when questioned about inventing new products, processes, or services; conducting initial R&D; creating internal tools or processes to build or implement final products; or undertaking the risk and costs of making, selling, and marketing a commercial product, "in no category are patents reported to provide even 'moderate' incentive for any of the four entrepreneurial activities." *Id.* at 1285.

[208] *Id.* at 1271-72.

[209] *See id.* at 1286-87.

[210] Long, *supra* note 194, at 637.

[211] *See id.* at 661-62.

[212] *See, e.g.*, Joi Ito, *One Venture Capitalist's View on Software Patents*, JOIITO.COM (July 08, 2005, 3:48 AM), http://joi.ito.com/weblog/2005/07/08/one-venture-cap.html; Fred Wilson, *Fred Wilson: Software Patents Should Not Exist*, FASTER TIMES (June 3, 2011), http://www.thefastertimes.com/venturecapital/2011/06/03/fred-wilson-software-patents-should-not-exist.

www.manaraa.com

development of new and better products and services or more efficient production and delivery of existing goods and services. If the incentive is poorly tailored to the social benefit, we should be unsurprised that rather than spurring innovation, software patents introduce friction, making innovation more difficult and its products more costly to obtain. Thus we must examine the interaction between patent's structure as an incentive system and the characteristics of software engineering and development, its industry, and its technological products.[213]

The mechanics of incentives have been studied extensively in the context of organizational behavior and management strategy, where managers strive to set incentives for employees and contractors that optimize performance in relation to pay.[214] The chief challenge in the field is to strike a balance between providing employees with incentives to work toward management's goals and keeping the employees' exposure to risk within an acceptable range.[215] Thus the percentage of earnings reflected in salespeople's commissions, investment bankers' bonuses for fund performance, and law firm associates' bonuses for hours worked all require assessment of the correlation between this measure and the worker's productive effort and the incentive's accuracy as a measure of value to the firm.[216]

If we consider the patent system as an incentive contract, it is even less precise than already-problematic incentives like performance bonuses because patents are at best a second-order

---

[213] *See* SUZANNE SCOTCHMER, INNOVATION AND INCENTIVES 39 (2004).

[214] *See* PAUL MILGROM & JOHN ROBERTS, ECONOMICS, ORGANIZATION AND MANAGEMENT 208 (1992) (collecting references); Bengt Holmstrom & Paul Milgrom, *Multitask Principal-Agent Analyses: Incentive Contracts, Asset Ownership, and Job Design*, 7 J.L. ECON. & ORG. (SPECIAL ISSUE) 24 (1991).

[215] *See id.* at 208 ("When rewards are based on results, uncontrollable randomness in outcomes induces randomness in the employees' incomes."). Since there is an element of randomness in both the measurement of performance and the correlation of effort to results, basing the employees' pay on results would subject them to the full force of this randomness risk. Risk averse employees would choose other employment rather than risk losing a large part of their prospective pay to measurement error. Of course there is also a risk of overpayment, but as Kahneman et al. document, most people are more loss-averse than risk seeking. *See* Daniel Kahneman et al., *Anomalies: The Endowment Effect, Loss Aversion, and Status Quo Bias*, 5 J. ECON. PERSP., Winter 1991, at 202-03.

[216] Pay that reflected only value to the firm, entirely tied to a product's sales for example, would subject the salesperson to risks outside her control such as supply shortfalls. Pay tied to a measure of effort such as hours, may induce bleary-eyed nights at the office with diminishing marginal value to the firm. Of course reducing the risk to the actor in control of assets has its own dangers; among the factors in the financial crisis, many point to the highly leveraged position of hedge fund managers, gambling with customers' money and not their own.

www.manaraa.com

prediction of desired results. As a society we are trying to induce innovation—the delivery to market of fully-realized products. Rather than measure innovation directly, however, the patent system focuses on inventions as "constructively reduced to practice" by means of patent applications,[217] which may or may not correspond to usable innovations.

Economists and management scholars distinguish between invention and innovation. Invention is the spark of an idea, while "innovation means invention implemented and taken to market."[218] In order for the public to benefit from development, it must have not only a source of ideas but also an engine of implementation.[219] There is often not a clear break, either in the language or the technological process, between basic research and applied science, or between initial application and industrial-scale production. The end-points, however, are clearly distinct: the idea of a transistor can be part of a theoretical model, but only the manufactured component can actually perform switching functions.

This split is reflected in the law: patent does not protect "abstract ideas."[220] This limitation on statutory subject matter is "consistent with the notion that a patentable process must be 'new and useful.'"[221] Abstract ideas are excluded because they are not directly useful. Machlup concluded that "[patent] incentives are supposed to generate technological inventions plus innovations—innovation being the first commercial application of a new idea. Invention without application is

---

[217] *See* U.S. PATENT & TRADEMARK OFFICE, MANUAL OF PATENT EXAMINING PROCEDURE § 2138.05 (2001), *available at* http://www.uspto.gov/web/offices/pac/mpep/documents/2100_2138_05.htm ("Reduction to practice may be an actual reduction or a constructive reduction to practice which occurs when a patent application on the claimed invention is filed. The filing of a patent application serves as conception and constructive reduction to practice of the subject matter described in the application. Thus the inventor need not provide evidence of either conception or actual reduction to practice when relying on the content of the patent application." (citing Hyatt v. Boone, 146 F.3d 1348, 1352 (Fed. Cir. 1998))).

[218] CHESBROUGH, *supra* note 90, at ix.

[219] Scotchmer describes economic theories of "production-function" and "endogenous-growth," differing in their assessment of the bottleneck's relationship to innovation and growth, and then sets up a third "ideas model" that needs funding both as a spur to the creation of ideas and to their further development as products. SCOTCHMER, *supra* note 213, at 54-55.

[220] *See* Bilski v. Kappos, 130 S. Ct. 3218, 3221 (2010) (denying patent to method for hedging risk) ("The Court's precedents provide three specific exceptions to § 101's broad [patent-eligibility] principles: 'laws of nature, physical phenomena, and abstract ideas.'" (quoting Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980))); *see also generally* BEN KLEMENS, MATH YOU CAN'T USE: PATENTS, COPYRIGHT, AND SOFTWARE (2006).

[221] *See Bilski*, 130 S. Ct. at 3225.

www.manaraa.com

useless; practical application may depend on patent protection even where invention does not."[222]

To be a socially useful incentive, then, patents must encourage innovation: the development and deployment of invention.[223] In software, however, the long road from idea to implementation often snags on patents early in the course. Engineers can describe what they want software to do—in terms that have been sufficient for the PTO—well before they have made it work. Pressures to patent early produce a thicket of pre-implementation claims.

The costs of the incentive's mismatch escalate as we consider patent's side-effects.[224] The old management saw, "what gets measured, gets done" applies to patents too. As patents appear to offer an objective measure of employees' productivity, companies offer bonuses to those who obtain patents, fueling a race to patent even more.[225] James Gosling, the computer scientist who invented and developed the Java programming language and left Sun Microsystems after its acquisition by Oracle, describes the extreme lengths to which the race can go: "Even though we had a basic distaste for patents, the game is what it is, and patents are essential in modern corporations, if only as a defensive measure. There was even an unofficial competition to see who could get the goofiest patent through the system."[226]

This response is common among technology companies that offer patent-filing bonuses or special recognition to inventors when patents are filed and issued. It also illustrates Milgrom and Roberts's "equal compensation principle": if there

---

[222] *See* S. SUBCOMM., ECONOMIC REVIEW, *supra* note 1, at 78.

[223] We may get innovation from rewarding invention, or find that innovation correlates with the visible invention, but we lose information along the way.

[224] The simple response, that patents improve efficiency by granting property rights to enable localized control of resource allocation, makes the same category error: the property right is only efficient if the right thing is propertized. If the lines are drawn too broadly or too early, transaction costs and hold-up swallow the gains.

[225] Richard S. Gruner, *Corporate Patents: Optimizing Organizational Responses to Innovation Opportunities and Invention Discoveries*, 10 MARQ. INTELL. PROP. L. REV. 1, 30 (2006).

[226] *See* James Gosling, *Quite the Firestorm: On a New Road*, NIGHTHACKS.COM (Aug. 15, 2010), http://nighthacks.com/roller/jag/entry/quite_the_firestorm ("In Sun's early history, we didn't think much of patents. While there's a kernel of good sense in the reasoning for patents, the system itself has gotten goofy. Sun didn't file many patents initially. But then we got sued by IBM for violating the 'RISC patent'—a patent that essentially said 'if you make something simpler, it'll go faster[.]' Seemed like a blindingly obvious notion that shouldn't have been patentable, but we got sued, and lost. The penalty was *huge*. Nearly put us out of business. We survived, but to help protect us from future suits we went on a patenting binge.").

www.manaraa.com

are several things an agent is expected to do, either all should be the subjects of similar incentive payments—or none should be.[227] Otherwise, the differential availability of incentive pay will distort the selection among activities—people will do most of what pays them the highest bonus.[228] What holds for individuals can apply to companies as well. The availability of patent, and the perceived benefits thereof, steers them toward patentable effort and away from other economically valuable activity.[229]

### *B. Software's Salient Distinctions*

Software-based industries share common features distinct from many other fields in which patenting has been studied in detail.[230] These include lower capital costs, which make the transaction costs of licensing a larger percentage; more smaller, distributed players; lower pure research costs and a focus on implementation; and lower uncertainty of development (as distinct from high uncertainty of market success); high complexity of a software product; and information costs and the nature of the software input.[231] Other differences apply more directly to the patent process, including the relative recency of the field and dearth of formal publication, which hinder prior art searches and novelty/non-obviousness assessments as well as lower patent's utility for information-transfer.

Software's structural distinctions affect both the operation of patent law and its incentive effect. The first set of problems has attracted more scholarly attention than the second.[232] In particular, the distributed, decentralized, and even atomistic nature of development makes the novelty and non-obviousness inquiries more difficult.[233] Much of software's prior

---

[227] *Cf.* MILGROM & ROBERTS, *supra* note 214, at 229.

[228] *See id.*

[229] *See* Parchomovsky & Wagner, *supra* note 137, at 19.

[230] *See supra* Part II. Many different kinds of firms make and use software. Some are pure "software" businesses, mass-market or customize; others make software to use in their research or business, to complement their hardware, to sell as integrated "solutions." Among them are many different modes of producing software: open source, shared, or proprietary. The heterogeneity still leaves significant commonality distinct from other fields.

[231] *See supra* Parts II.A-II.D.

[232] Even that comparison is relative. Cohen and Lemley complain that "[w]hile there is a voluminous literature on whether software is (or should be) patentable subject matter, there is much less discussion of other patent validity issues." Cohen & Lemley, *supra* note 201, at 13-14.

[233] *See supra* Part II.

www.manaraa.com

art is published in ephemeral Internet forum posts and mailing lists, or in the source code of fleetingly available commercial products; the prior art is not, however, published in the PTO's database of patents or recognized journals.[234] That challenge is intensified by the transitional state of the patent system with regard to software, which has still only recently been recognized as patentable subject-matter, so the PTO's experience with the field and its databases are thinner than those in non-software fields.[235]

Disclosure and enablement tend to be stinted as well. The patent's disclosure is not of functioning code, but of procedural steps.[236] Another person "of skill in the art" could implement from these outlines, but without saving much time following the originator.[237] Moreover, because of the abstraction of many software patent claims and the wide breadth given to equivalents, it can be difficult to locate a patent on a given subject or to glean from its claims and disclosure any indication as to what the patent will be judged to cover. Is it worth spending more time trying to fix these examination and scope problems? Mark Lemley's *Rational Ignorance* argues that the answer is no because most patents are not worth more time and expense in examination.[238] Yet while many bad patents will never be asserted against anyone, software patents present a likely scenario in which "bad" patents on basic functions are used broadly.

The structural differences in software patent's incentives have been less well-studied. In comparison to pharmaceutical and semiconductor industries, software's capital costs are lower, primarily because its R&D is structured differently.[239] In accord with Moore's Law, the costs of computer hardware, the chief capital input to software development, have decreased steadily.[240] Whereas the capital and regulatory

---

[234] *See* Cohen & Lemley, *supra* note 201, at 13 ("Unlike inventions in more established engineering fields, most software inventions are not described in published journals.").

[235] Bradford L. Smith & Susan O. Mann, *Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?*, 71 U. CHI. L. REV. 241, 258 (2004).

[236] *See supra* note 65 and accompanying text.

[237] *See supra* Part I.C.

[238] *See* Lemley, *supra* note 190, at 1496-97.

[239] *See supra* note 153 and accompanying text.

[240] "Moore's Law" derives from a prediction by Intel CEO Gordon Moore that transistors' counts on an integrated circuit (and thus processing power) would double every two years. That prediction has continued to hold over more general measures of computing power, doubling approximately every 18 months and its associated costs decreasing. Paul. E. Ceruzzi, *Moore's Law and Technological Determinism: Reflections*

www.manaraa.com

requirements of drug development tend to cause consolidation in pharmaceutical research among a relatively small number of large companies, software exhibits no such pressure. Hence there are more independent developers who can experiment without having to convince others in advance. An individual or small group can aggregate the resources for software development independently and thus does not have to recruit outside investors with the promise of a privileged stake in exclusive rights in order to develop at least as far as the proof-of-concept stage.[241] Thus the archetypal software innovator, the "guy or gal in the garage," is actually a significant presence because a garage (or loft, or cheap co-working space) can hold the relatively ubiquitous tools of software development—the computer and its network connection—fairly cheaply and remain well within the resources available to the independent developer.

Nor does a company need a huge staff to develop software. A small team or single programmer can complete significant projects. Indeed, Frederick Brooks famously noted that "[a]dding manpower to a late software project makes it later," finding that coordination problems among larger teams swamped the added productivity of extra workers.[242] While there is some specialization (e.g. user-interface design), many software projects can be undertaken by the same programmers from start to finish, without task-based division of labor.[243] Modular design—reducing the interdependencies that exist among software components—can only partially reduce these coordination challenges; modularity often waits for a second-round design.[244]

---

*on the History of Technology*, 46 TECH. & CULTURE 584, 585 (2005); Michael Kanellos, *Moore's Law to Roll on for Another Decade*, CNET.COM (Feb. 10, 2003), *available at* http://news.cnet.com/2100-1001-984051.html; BURK & LEMLEY, *supra* note 149, at 83.

[241] For Barbara van Schewick's description of distributed innovation and its value on the Internet, see BARBARA VAN SCHEWICK, INTERNET ARCHITECTURE AND INNOVATION 347 (2010).

[242] BROOKS, *supra* note 98, at 25.

[243] *See, e.g.*, NATHAN DENNY ET AL., NEXUS OF ENTREPRENEURSHIP & TECH., UNIV. ARIZ., AGILE SOFTWARE FOR THE 24-HOUR KNOWLEDGE FACTORY ENVIRONMENT, *available at* http://next.eller.arizona.edu/publications/ssrn/AlternativeLinks/Agile%20Software%20Processes%20for%20the%2024-Hour%20Knowledge%20Factory%20Environment.pdf. Agile programming methodologies specifically call upon the same programmers to run through various tasks of design, implementation, and testing. *See generally* ALISTAIR COCKBURN, AGILE SOFTWARE DEVELOPMENT (2001).

[244] *See* CARLISS Y. BALDWIN & KIM B. CLARK, DESIGN RULES, THE POWER OF MODULARITY 355-57 (2000).

www.manaraa.com

These minimal capital requirements allow software development to be widely dispersed, conducted by both specialists and non-specialists, big and small firms, and companies and individuals. There is no single natural clearinghouse for information exchange—either for sharing of research or for licensing of patents. This does not mean information is not shared—many programmers participate in informal information exchange through forums and mailing lists, share demonstrations, and read professional publications.[245] Open source and Free Software development exhibit an extreme version of information-sharing: much development, even by contributors from large firms, is done on an individual basis, with a great deal of information published along the way.[246] Propertizing and pricing information, rather than bringing efficiency to its transfer, burdens the information exchange. Against this baseline, demands for patent license fees can be a significant burden, even if the fees are low.[247]

In combination, we see the full force of software patents' harm, as they contribute to the prevalence of trolls, thickets, and pools impeding software innovation.

1. Trolls[248]

Because of low barriers to entry, the software industry is rife with start-ups: entrepreneurial ventures and individuals with ideas. Some are successful, while others inevitably fail.[249] Since patents correlate poorly with product success, many unsuccessful companies nonetheless own issued patents. Patent protects the idea, as "constructively reduced to practice" by the filing of a patent application.[250] If patents are seen as valuable signals, then, would-be venture-funded firms participate in the rush to the patent office to get theirs, but the first in line will not have necessarily made a functioning prototype, much less a deployed system. Another developer who has developed a functioning system, on the other hand, is

---

[245] *See, e.g.*, DEV SHED FORUMS, http://forums.devshed.com (last visited Feb. 2, 2013); CODECALL, http://codecall.net (last visited Feb. 2, 2013); CODEGURU, http://codeguru.com (last visited Feb. 2, 2012).

[246] *See supra* Part II.D.

[247] *See* Kapczynski, *supra* note 88, at 993-94.

[248] *See infra* note 252.

[249] *See* Rita Gunther McGrath, *Falling Forward: Real Options Reasoning and Entrepreneurial Failure*, 24 ACAD. MGMT. REV., Jan. 1999, at 13-30 (characterizing the role of failure in entrepreneurial innovation).

[250] *See* 37 C.F.R. § 1.131(b) (2012).

www.manaraa.com

unlikely to have learned it from the patent literature. Success in the market may be determined by factors of marketing strategy, network effects, and may be only loosely coupled with the pure quality of the software product. In sum, patent does not predict quality.[251]

This mismatch helps to produce patent trolls.[252] In many instances, a company will have filed one or more software patents, but then the company fails because its product never reached market, failed to work, or because the market just did not like it. It is also possible that the patent, obtained early, distracted engineering, strategic, and financial resources from core business. Still, failure should be acceptable as an expected part of the entrepreneurial system; just as innovation happens through risk-taking, the entrepreneurial society is able to forget failures and allow its founders to move on.[253]

But, the patent does not die with the company. Instead, it persists, left like a land-mine long after that particular company has departed the battlefield. Often, "intellectual property," the only asset left, gets sold off to a holding company willing to pay something for this hunting license—a patent troll who then proceeds to look for others successfully making or using similar technology.[254] By protecting early-stage ideas, the patent system has fed this pipeline, inducing not productive innovation but unproductive races for merely redistributive ends.

"Troll" may be an epithet, but "non-practicing entities" (NPEs) share a set of distinguishing features. In particular, they contribute nothing to the development and deployment of technology.[255] They maintain their patents solely for the purpose of perfecting claims for a share of the profits of others who actually make things—claims asserted in legal threats and

---

[251] *See supra* note 59; *see also* Louis Kaplow, *The Patent-Antitrust Intersection: A Reappraisal*, 97 HARV. L. REV. 1813, 1823 (1984).

[252] Trolls, or "non-practicing entities," can threaten and sue based on the patents in their portfolios, but cannot be counter-sued for patent infringement (as is common in suits between competitors) because the trolls are not practicing any of the arts they seek to license; they only exist to extract royalty revenue. Colleen V. Chien, *Of Trolls, Davids, Goliaths, and Kings: Narratives and Evidence in the Litigation of High-Tech Patents*, 87 N.C. L. REV. 1571, 1578-80 (2009).

[253] *See* Sankaran Venkataraman, *Regional Transformation Through Technological Entrepreneurship*, 19 J. BUS. VENTURING 153, 164-65 (2004) (describing the role of social safety nets and acceptance of failure to entrepreneurial productivity).

[254] *See supra* note 197; *see also* Gerard N. Magliocca, *Blackberries and Barnyards: Patent Trolls and the Perils of Innovation*, 82 NOTRE DAME L. REV. 1809, 1817 (2006–2007).

[255] James Bessen et al., *The Private and Social Costs of Patent Trolls*, REG., Winter 2011–2012, at 26; John M Golden, *"Patent Trolls" and Patent Remedies*, 85 TEX L. REV. 2111, 2112 (2007).

www.manaraa.com

costly litigation against manufacturing entities.[256] NPEs are not sensitive to the same forces as going concerns. When they explicitly monetize the process, they can crowd out non-monetary considerations in innovation.

Defenders of non-practicing entities typically argue that they provide liquidity in patent markets[257]: "Liquidity" is only valuable if it provides resources for something independently valuable. If software patents serve as stumbling blocks rather than building blocks for future innovators, providing "liquid" encouragement to those who produce them is counterproductive.

2. Thickets

Since each patent covers only a small component of a program's overall function, any single patent can block; many patents pooled together, however, can block with real market power. It takes only a few patents stacked together to consume the entire profit of a software product, given that "reasonable royalties" are taken as percentages for any patent serving as a necessary contribution.[258] Moreover, because software components are not purchased and exhausted in the same way that hardware components may be, it is riskier for downstream innovators to build upon existing work.[259]

Since independent invention is not a defense, a developer may unwittingly stumble across another's patents, which can then become too costly to clear. In this context, patents serve an anti-coordination function.[260] Moreover, software products are complex, incorporating many functional components, whether separated along modular lines or fused together. Patents can be written to many sizes: if small, then

---

[256] Bessen et al., *supra* note 255, at 26, 34-35.

[257] *See* Michael Risch, *Patent Troll Myths*, 42 SETON HALL L. REV. 457, 459, 466 (2012); Sannu K. Shrestha, Note, *Trolls or Market-Makers? An Empirical Analysis of Nonpracticing Entities*, 110 COLUM. L. REV. 114, 115-16, 130 (2010).

[258] *See generally* Mark A. Lemley & Carl Shapiro, *Patent Holdup and Royalty Stacking*, 85 TEX. L. REV. 1991 (2007).

[259] *See id.* Someone who invests in building on an unpatented commodity can purchase components on the open market and thus expect to reap all the rewards of success or losses of failure, while someone who builds on a patented product may still bear all the risks, but face royalty demands if he succeeds. It may be argued that the switch from sale to licensing of components is beneficial price discrimination, which enables component-makers to reap the full value of *their* inventions. Much finer price-discrimination can deprive the downstream users of spillover benefits that would have given them an opportunity to innovate.

[260] *See* Michael A. Heller, *The Tragedy of the Anticommons: Property in the Transition from Marx to Markets*, 111 HARV. L. REV. 621 (1998).

www.manaraa.com

numerous potential patents exist within a given product; if large, then there will be many different ways to accomplish the patent's ends, and the scope of equivalents will be too large.

3. Pools

At other times, patent pools are used to coordinate, or standardize, an industry. Pooling can serve to clear the thickets of competing claim, by drawing all of those with claims to a technology into one place for joint licensing.[261] Pools often cross-license freely to anyone who contributes an "essential" patent—providing added inducement for companies to obtain such patents—but this creates problems of a different sort for those outside the pool or to those for whom patents on per-copy royalty terms are not an option, such as Free Software.[262]

Wagner and Parchomovsky see the centralizing effects of portfolio use, but they do not pursue its full implications, that is, when incumbents coordinate in pools to erect barriers to entry against newcomers.[263] That centralization of innovation opportunities restricts who can develop new technology.[264] Such control reduces options for diversity of experimentation and disruptive innovation.[265]

Further, patents pose a particular challenge to open source and Free Software that is developed collaboratively outside of traditional firms.[266] Apart from the already-noted mismatch between patent fees and low-cost or free distribution, per-copy royalties and usage limits are incompatible with the freedom to modify and redistribute that lies at the heart of

---

[261] Philip B. Nelson, *Patent Pools: An Economic Assessment of Current Law and Policy*, 38 RUTGERS L.J. 539, 539 (2007).

[262] *Id.* at 542.

[263] *See* Parchomovsky & Wagner, *supra* note 137, at 66-67 ("As firms increasingly use portfolios as ever more effective tools for the domination of innovation markets, the results would seem to be (1) a broad consolidation and centralization of inventive activity within large firms or groups of firms organized around jointly developed patent portfolios, and (2) the use of portfolios to achieve real market power or otherwise cartelize markets." (footnotes omitted)).

[264] Compare this with the centralization by anticircumvention measures and their licensing described in Wendy Seltzer, *The Imperfect Is the Enemy of the Good: Anticircumvention Versus Open User Innovation*, 25 BERKELEY TECH. L.J. 909, 958 (2010); *see also* VAN SCHEWICK, *supra* note 241, at 342 (describing the benefits of an architecture that enables distributed innovation, permitting experimentation at the edges to succeed).

[265] *See generally* CLAYTON M. CHRISTENSEN, THE INNOVATOR'S DILEMMA (2003).

[266] *See* BENKLER, *supra* note 176, at 63-64; Seltzer, *supra* note 264, at 965-66 (describing free and open source software and the welfare benefits of open user innovation).

www.manaraa.com

Free Software.[267] The social costs of the patent incentive are high and mounting.

### *C. Appropriability Alternatives*

A more sophisticated argument for patents in early-stage ideas develops a more complex theory of causation. It is not that we need incentives for ideas but that we want to give inventors "appropriability strategies" for their information goods, and we choose to do that through early-stage patents for purposes of administrability.[268] Given the negative side effects of this appropriation strategy, however, we must also consider how alternatives compare. Software businesses need not resort to patent, as software offers a wealth of alternative appropriability strategies: copyright, trade secret, first-mover advantage, network effects, and complements.

#### 1. Non-Patent Intellectual Property

*Copyright*. Copyright protects software against direct copying of source or compiled binary programs. A fully constructed program is protected by copyright against reproduction, including reproduction in the course of use.[269] While copyright protects only the expressive elements of software, and not its functions, this separation preserves the first-mover's lead-time advantage. A competitor may reverse engineer in order to copy software's function,[270] but copyright-safe reverse-engineering is time-consuming.[271] To guard against claims of copyright infringement, a competitor will have one

---

[267] *See* Free Software Found., Inc., *GNU General Public License* v.3 (June 29, 2007), https://www.gnu.org/copyleft/gpl.html.

[268] *See* SCOTCHMER, *supra* note 213, at 2-3.

[269] *See* MAI Sys. Corp. v. Peak Comp., Inc., 991 F.2d 511, 518-19 (9th Cir. 1993) (holding that loading a copy of a computer program into RAM constituted "reproduction" under the Copyright Act). Section 117 creates only narrow statutory exceptions that authorize such reproduction only by the lawful owner of a computer program or by maintenance on a machine with an authorized copy. *Id.* at 519 n.6.

[270] *See* Pamela Samuelson & Suzanne Scotchmer, *The Law and Economics of Reverse Engineering*, 111 YALE L.J. 1575, 1608 (2002). *Compare* Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1237 (3d Cir. 1986) (holding that along with its literal expression, a program's "structure, sequence, and operation" are protected by copyright), *with* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1518 (9th Cir. 1992) (holding copying of the object code necessary to the process of disassembly and re-creation of the functionality to be fair use); *see also* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995), *aff'd*, 140 F.3d 70 (1st Cir. 1998) (permitting copying of functional elements of user interface and menus as "method of operation.").

[271] *See Whelan Assocs*., 797 F.2d at 1237 (noting that it would take a competitor almost as long to reverse engineer).

www.manaraa.com

team examine or decompile the original program and pass only functional descriptions to a second team of programmers in a "clean room" for re-implementation. "Reverse engineering of object code is generally so difficult, time-consuming, and resource-intensive that it is not an efficient way to develop competing but nonidentical programs."[272] Moreover, reverse engineering does not short-cut the time-consuming steps of debugging and testing. To re-implement successfully, developers will have to rewrite and test their new programs and, if trying to achieve compatibility with an existing program, will have to preserve not only features but bug-compatibility.[273]

*Trade Secrecy*. Many software vendors likewise exploit trade secrecy. Those who distribute software in object code form can hide the details of its construction and modification, sometimes with further layers of obfuscation beyond mere compilation.[274] While reverse engineering is a legitimate means of acquiring trade secrets, producers are under no obligation to make reverse engineering easy. Unlike in most other areas, trade secret and copyright can coexist in software. The copyright office, which keeps public records of copyright registrations, accepts redacted deposits of program code, enabling developers to register while still keeping the source code to their work secret.[275] Trade secrecy also bars commercial fraud, such as taking programmatic details from a competitor or former employer. Moreover, producers can achieve greater secrecy in software that is not distributed to end-users but offered as "software as a service," for example, by running on a producer-operated server to which users connect.[276] Even less of the functional detail is exposed to end-users, who can analyze its function only through inputs and outputs.[277]

---

[272] *See* Samuelson & Scotchmer, *supra* note 270, at 1613 (citing Andrew Johnson-Laird, *Software Reverse-Engineering in the Real World*, 19 U. DAYTON L. REV. 843, 843 (1994)).

[273] *Id.* at 1613-14.

[274] *See, e.g.*, Computer Object Code Obfuscation Using Boot Installation, U.S. Pat. App. 2009/0235089, *available at* http://appft.uspto.gov/netahtml/PTO/srchnum.html (search "20090235089").

[275] *See* U.S. COPYRIGHT OFFICE, CIRCULAR 61.0812, COPYRIGHT REGISTRATION FOR COMPUTER PROGRAMS (2012), *available at* http://www.copyright.gov/circs/circ61.pdf (instructing submitters to deposit computer programs "with trade secret portions blocked out").

[276] *See generally* Press Release, Gartner, Inc., Gartner Says Worldwide Software as a Service Revenue Is Forecast to Grow 21 Percent in 2011 (July 7, 2011), *available at* http://www.gartner.com/newsroom/id/1739214.

[277] It is possible to reverse-engineer such a black-box, but it is more difficult. *See generally, e.g.*, ANDREW "BUNNIE" HUANG, HACKING THE X-BOX: AN INTRODUCTION TO REVERSE ENGINEERING (2003).

www.manaraa.com

Finally, consider patent as a response to Arrow's information "paradox": the purchaser cannot know information's value until it has been disclosed, at which point (without intellectual property's constraints) the buyer would have no need to pay.[278] The hypothesis behind this theory is that without patent, parties would not disclose their ideas, either as products on the market to end-users or as ideas-in-development in search of funders and collaborators. Instead, against the threat from industrial competition, inventors might integrate vertically to keep trade secrets within the expanded firm. Against the consumer, they would try to obfuscate operation to preserve secrecy against reverse engineers.

While these closure reactions may well reduce opportunities for innovation *on those elements the company can shield from disclosure*, they do not have the infectious reach of patents. Patents can intrude even on independent development since, in contrast to copyright, independent invention is not a defense to patent infringement claims.[279]

2. Business Strategy

The wealth of literature on software industry business strategies mentions intellectual property only fleetingly, spending more time on platform management, establishing first-mover advantage and network barriers to entry, and selling complementary goods or services.[280] That, and the success of the industry in the years before software patents

---

[278] *See* KENNETH J. ARROW, ESSAYS IN THE THEORY OF RISK-BEARING 152 (1971) ("[Information's] value for the purchaser is not known until he has the information, but then he has in effect acquired it without cost.").

[279] Copyright applies only to acts of copying and does not bar independent discovery. *See* Sheldon v. MGM Pictures Corp., 81 F.2d 49, 54 (2d Cir. 1936). As Learned Hand put it, "[I]f by some magic a man who had never known it were to compose anew Keats's Ode on a Grecian Urn, he would be an 'author,' and if he copyrighted it, others might not copy that poem, though they might of course copy Keats's." *Id.* By contrast, independent invention is not a defense to patent infringement, 35 U.S.C. § 271(a) (2006), despite the frequency of concurrent invention in history (The steam engine and telephone are early famous examples.).

[280] *See*, *e.g.*, MICHAEL A. CUSUMANO, THE BUSINESS OF SOFTWARE (2004); DAVID S. EVANS ET AL., INVISIBLE ENGINES: HOW SOFTWARE PLATFORMS DRIVE INNOVATION AND TRANSFORM INDUSTRIES (2006), *available at* http://mitpress.mit.edu/catalog/item/ebook.asp?ttype=2&tid=11447; ANNABELLE GAWER, PLATFORMS, MARKETS AND INNOVATION (2010); ANNABELLE GAWER & MICHAEL A. CUSUMANO, PLATFORM LEADERSHIP: HOW INTEL, MICROSOFT, AND CISCO DRIVE INDUSTRY INNOVATION (2002); Jean-Charles Rochet & Jean Tirole, *Platform Competition in Two-sided Markets*, 1 J. EUR. ECON. ASS'N 990 (2003); Annabelle Gawer, Towards a General Theory of Technological Platforms (2010) (unpublished manuscript), *available at* http://www2.druid.dk/conferences/viewpaper.php?id=501981&cf=43.

www.manaraa.com

were seen as viable, argue strongly against their value now. Software producers can appropriate value to their efforts from strategies involving marketing or business structures.

*First-Mover Advantage*. Simply being first gives an innovator the opportunity to establish itself as category leader, where it may capture an audience, lock-in resources, or secure complements and make them unavailable to imitators.[281]

*Complements*. Some producers deploy software as a complement to hardware or services. If the software is most useful when coupled with proprietary hardware on which the producer profits, or if it adds value to that hardware, the producer may be able to give the software away (or sell it cheaply) while profiting on the hardware. The software can even serve as an advertisement of the hardware's capabilities.[282] IBM's contributions to the open source Linux kernel enhance the support for its specialized hardware and the performance of systems it sells.[283] Some producers go further and lock their software to hardware, exploiting the DMCA's anticircumvention provisions to bind the two together.[284] In a complements market, where both components are necessary, it does not matter on which component the producer takes profit. (Consider the complementary markets for left and right shoes. A monopolist on left shoes could give away right shoes while reaping supra-competitive prices on the matching left shoes.) Thus, where it is difficult to profit from software directly, producers may bundle its price into something else that is more easily metered.

*Network Effects*. Some software producers exploit network effects in use. That is, one user's use of software adds value for others doing so.[285] Like the telephone or fax machine, a software-service like Facebook or Twitter exhibits such network effects: users' utility depends on being able to connect with their friends using the same services. Other software,

---

[281] *See generally* Marvin B. Lieberman & David B. Montgomery, *First-Mover Advantages*, 9 STRATEGIC MGMT. J. 41, 41-47 (1988).

[282] Thus, accelerated video cards often come with bundled graphics or videogame applications to show off the cards' power. *See, e.g.*, Dean Takahashi, *AMD Bundles Marquee PC Games with Its Radeon Graphics Cards*, VENTUREBEAT.COM (Oct. 21, 2012, 9:01 PM), http://venturebeat.com/2012/10/21/amd-bundles-marquee-pc-games-with-its-radeon-graphics-cards/.

[283] *See* David Berlind, *Open Source: IBM's Deadly Weapon*, ZDNET.COM (Apr. 8, 2002, 7:00 PM), http://www.zdnet.com/news/open-source-ibms-deadly-weapon/296366.

[284] *See* Apple Inc. v. Psystar Corp., 658 F.3d 1150, 1153 (9th Cir. 2011) (describing Apple's use-licensing and technological measures "to prevent Mac OS X from operating on non-Apple computers," and finding Psystar in violation of copyright and anticircumvention laws).

[285] *See* SHAPIRO & VARIAN, *supra* note 57, at 174.

www.manaraa.com

such as word processing and graphics tools, shows "virtual" network effects: users generally use it in solitary mode but seek to exchange files and formats with others and benefit when others are using compatible or the same software. Network effects can contribute to locking-in customers, inducing them to continue using the same products or upgrades even if others (such as imitators) would serve their purposes equally well absent network externalities.[286]

*Platforms*. Software can create a platform on which goods or services can be exchanged. Microsoft's Windows 7 and Apple's MacOS operating systems are platforms for application software. Google and Facebook are platforms for advertising to end-users. In a "multi-sided market," the platform proprietor may subsidize one side of the market in exchange for extracting greater revenues elsewhere.[287] Thus, Google and Facebook provide software-services free to the end-users conducting searches or networking with friends, while charging advertisers to reach those end-users.[288] The advertisers are buying access to the crowds more than to particular software algorithms.

*Standards*. De facto or de jure standards affect the network.[289] They can promote interoperability, or, when patented, can give control of the network value to a patent-holder, who may be responsible for only a small fraction of it.[290] Patents may promote coordination among those with compatible business models, while excluding those whose mode of operation would not generate per-copy royalties.

*Services*. Software development is often sold as a service, through consulting contracts or employment. The success of an existing product or component advertises developers' services to others who may be interested in new development or customization.[291] The utility of already deployed software can serve as more reliable signals than the number of patents.

---

[286] *See id.* at 184-85; *but see* S.J. Liebowitz & Stephen E. Margolis, *Path Dependence, Lock-in, and History*, 11 J.L. ECON. & ORG. 205 (1995).

[287] *See* Thomas Eisenmann et al., *Strategies for Two-Sided Markets*, HARV. BUS. REV., Oct. 1, 2006, at 94.

[288] *See* Rafe Needleman, *Facebook vs. Google: The Epic Battle*, CNET (May 17, 2012, 5:55 AM), http://news.cnet.com/8301-1023_3-57435965-93/facebook-vs-google-the-epic-battle (describing business models and revenue sources of both firms).

[289] *See* S.J. Liebowitz & Stephen E. Margolis, *Should Technology Choice Be a Concern of Antitrust Policy*, 9 HARV. J.L. & TECH. 283, 290-91 (1996); *see also* SHAPIRO & VARIAN, *supra* note 57, at 237-38.

[290] *Cf.* Mark A. Lemley & David McGowan, *Could Java Change Everything? The Competition Propriety of a Proprietary Standard*, 43 ANTITRUST BULL. 715 (1998).

[291] *See* Karim R. Lakhani & Eric von Hippel, *How Open Source Software Works: "Free" User-to-User Assistance*, 32 RES. POL'Y 923, 923 (2003); Lakhani & Wolf,

www.manaraa.com

Alternative appropriability regimes provide incentives for software development, while sparing the "embarrassment"[292] of patent's exclusive rights to ideas. These regimes operate under different constraints than the patent system. In particular, in order for any of the methods described above to appropriate returns to a software developer, invention must be taken beyond the mere idea point to achieve implementation. Furthermore, many of these regimes provide appropriability without exclusive rights to the underlying concepts. For example, under copyright protection, interoperable software programs coexist; improvement is not restricted to the initial developer. Some of these strategies tend toward monopoly, particularly those involving network effects and platforms. The length of those monopolies, however, will be set by the market, with competitors forcing even platform providers to innovate[293] (and possible regulatory and judicial intervention where the market fails),[294] rather than by the fixed term of patent.

Most important, these appropriability alternatives do not bar independent exploration of the problem space. They do not guarantee that one who explores and develops new software products will gain a property right, but they do assure that the explorer will not stumble across others' unmarked property along the way.[295]

## V. RESHAPING THE LAW

The challenge of reshaping the law is multi-faceted, involving choosing the right law and policy for software innovation, line-drawing to set the right granularity for its application, and getting that law enacted. Focusing on any

---

*supra* note 121, at 3 (describing app and game mod competitions as job fairs, and open source participation as part of a resume); *see generally* CHRIS ANDERSON, FREE: THE FUTURE OF A RADICAL PRICE (2009).

[292] *See* THOMAS JEFFERSON, *Letter from Thomas Jefferson to Isaac M'Pherson Aug. 13, 1813*, *in* BASIC WRITINGS OF THOMAS JEFFERSON 708, 712-13 (Philip S. Foner ed., 1944).

[293] *See* Michael L. Katz & Howard A. Shelanski, *"Schumpeterian" Competition and Antitrust Policy in High-Tech Markets*, COMPETITION 4 (2005) (describing a dynamic in which "firms do not compete simultaneously for a share of the market, but rather sequentially for the market as a whole"). A platform provider who becomes complacent risks losing even the momentum it has built.

[294] *See, e.g.*, United States v. Microsoft Corp., 253 F.3d 34 (D.C. Cir. 2001).

[295] *See generally* Stephen M. Maurer & Suzanne Scotchmer, *The Independent-Invention Defence in Intellectual Property*, 69 ECONOMICA 535 (2002) (making an economic theory case for an independent invention defense: it will tend to lower the rents from patent's exclusive right, and will thereby make the pre-patent race less wasteful).

www.manaraa.com

single level will produce ineffective, impractical, or unimplementable recommendations.

If software patents are counter-productive—and we would foster more innovative activity and product deployment without these impediments—it is still no simple matter to change the legal treatment of software inventions. In Congress, proponents of patent reform began with grand plans for what became the "America Invents Act," including a restructuring of willful infringement to remove the threat of treble damages that stops many engineers from even reading patents. What eventually came out of negotiations was a pale shadow of these plans, making only minor changes to the law's structure and substance. Thus, although it might be productive to change the law to require that an implementation be submitted and published, rather than accepting its abstract description, the public choice challenges of tinkering with a mostly-acceptable status quo may prove too great. Can anything change now given the "installed base" of software patent defenders?

Moreover, even if stronger legal prohibitions existed than the current restriction on "abstract ideas," statutes would remain open to strategic loophole-finding. The European Union officially does not recognize or grant software patents, but plenty of software is patented on the fiction that its implementation is "on a machine" that contains more than just software.[296]

Instead, since courts will inevitably play a necessary role in implementing software patent reform, they could use the leeway in existing statute and precedent to swing away from protecting software. The courts could, therefore, correctly recognize software as a collection of mere abstract ideas implemented on general-purpose machines.[297]

Technological specificity in the patent system introduces transaction and information costs, as participants try to game their way into the most protective category—even if they would all be better off in a scenario in which the category protected none of them. On the other hand, a uniform patent system has its own costs, because uniform rules may fit none of the categories well. As we have seen, what works for pharmaceuticals and large-scale manufacturing does not fit the dynamics of the software industry.

---

[296] Susan J. Marsnik & Robert E. Thomas, *Drawing a Line in the Patent Subject-Matter Sands: Does Europe Provide a Solution to the Software and Business Method Patent Problem?*, 34 B.C. INT'L & COMP. L. REV. 227, 277 (2011).

[297] *See* BURK & LEMLEY, *supra* note 149, at 122-24.

www.manaraa.com

As well as addressing specific problems with patent theory and patent law as applied to the software industry, this analysis points to larger challenges in the "stickiness" of property rules. Not only are individual endowments of property and property rights sticky,[298] but structures of property rules and their definitions of the underlying "property" persist even into contexts well beyond their original justifications.

Compare debates around wireless spectrum use. Auctions to establish property rights in frequency bands were initially seen as more efficient than command-and-control licensing for a particular use.[299] But once rights in those bands were established, rights-holders and their boundaries become difficult to dislodge, even for more productive re-deployment of the spectrum, such as from broadcast to "white-spaces" user-defined use.[300] Moreover, even if some participants could be induced to trade their rights if their counterparties could aggregate the value, uses that cut across property boundaries—such as spread-spectrum technology—have largely been unable to coordinate enough rights-holders and have faced hold-out problems.[301]

For all the academic articulation of problems with the patent system, patent law has proven remarkably resistant to reform. This property lock-in should give us pause in creating new rights and will thus require creative coalition-building, such as Creative Commons, to re-imagine existing grants.

---

[298] *See* Daniel Kahneman et al., *Experimental Tests of the Endowment Effect and the Coase Theorem*, 98 J. POL. ECON. 1325, 1345 (1990).

[299] *See* R.H. Coase, *The Federal Communications Commission*, 2 J.L. & ECON. 1, 20-21 (1959); *see generally* Harold Demsetz, *Industry Structure, Market Rivalry, and Public Policy*, 16 J.L. & ECON. 1 (1973).

[300] *See* David J. Farber & Gerald R. Faulhaber, *Spectrum Management: Property Rights, Markets, and the Commons* 9 (AEI-Brookings Joint Center for Regulatory Studies, Working Paper 02-12, 2002); Yochai Benkler, Open Wireless vs. Licensed Spectrum: Evidence from Market Adoption 19 (unpublished manuscript), *available at* http://www.benkler.org/Open_Wireless_V_Licensed_Spectrum_Market_Adoption_current.pdf.

[301] *See* Michael A. Heller, *The Tragedy of the Anticommons: Property in the Transition from Marx to Markets*, 111 HARV. L. REV. 621, 625 n.15 (1998).

www.manaraa.com