February 16, 2006

COMMERCIALIZING OPEN SOURCE SOFTWARE: DO PROPERTY RIGHTS STILL MATTER?

Ronald J. Mann*

Table of Contents

---

* Ben H. & Kitty King Powell Chair in Business and Commercial Law, Co-Director, Center for Law, Business & Economics, University of Texas School of Law. I thank Allison Mann for inspiration and wisdom and Ken Myers for excellent research assistance. I am grateful to executives at Codeweavers, Hewlett-Packard, IBM, Intel, JBoss, Matrix Partners, Microsoft, MontaVista, MySQL, Novell, the Open Source Development Laboratory, the Open Source Initiative, Pervasive, Red Hat, and Worldview Capital Partners, who took time from their schedules to speak with me about the ideas in this paper. I also thank the participants in the Intellectual Property workshop at Boalt Hall, for useful comments on an earlier draft.

Abstract

A major shift toward open source software is underway. Companies are more critically evaluating the cost effectiveness of their IT investments, seeing the benefits of collaborative development, and looking for ways to avoid vendor lock-in. At the same time, academics and industry visionaries are criticizing the use of a traditional appropriation mechanism for innovation—the patent—by bemoaning the decisions of U.S. and foreign governments to permit software patents, the rising numbers of patents on software-related innovations (the so-called "arms race" build-up), and the cost and frequency of patent litigation in the software industry. The critics generally have applauded the shift towards open source, albeit for somewhat varying reasons.

This paper responds to those trends by analyzing the role of property rights in the open source model, with a particular focus on the effectiveness of the appropriation mechanisms that the open source model uses in lieu of intellectual property rights. I make two main points. First, I argue that open source's commercial success is intertwined with its incorporation into traditional commercial value chains. What that means is that open source cannot continue to grow in commercial importance without the property rights that are necessary for firms to profit at other points of the value chain. Second, I argue that despite open source's distributed development process, open source in the real world is likely to support an increasing concentration in the software industry. The reason is that the proprietary firms best situated to exploit commercial interactions with open source will be large firms, particularly large services firms. Smaller firms will be less successful as services firms, and far less successful at exploiting the value-chain interactions that have driven commercial open source.

COMMERCIALIZING OPEN SOURCE SOFTWARE: DO PROPERTY RIGHTS STILL MATTER?

I. Introduction

For several years now, open source software products have been gaining prominence and market share. Yet it is not the products themselves that are provocative, but the way in which they are developed and distributed. Two related features of the model are distinctive: the use of collaborative development structures not contained within the boundaries of a single firm, and the lack of reliance on intellectual property rights as a means of appropriating the value of the underlying innovations. Firm-level control of property is replaced by a complex set of informal and (sometimes) contractual relations among strategic partners not joined by traditional firm boundaries. I argue here that those relations reflect not coalescence towards industry norms driven solely by superior output, but rather a series of strategic moves and countermoves that have had the effect of opening some markets but closing others, substantially reducing profit margins, and fostering consolidation of a traditionally fragmented industry.

I have written elsewhere about the role of intellectual property rights in proprietary models of software development, where intellectual property rights are used (albeit somewhat ineffectively) by firms to exploit the value of their internal R&D investments. In that work, I generally reject the idea that the sheer number of patents is creating a thicket that deters innovation, largely because of the positive evidence of a robust startup market and because of the lack of evidence of any chilling of investment. More generally, I have argued that many of the criticisms of software patents fail to account for the apparent benefits those patents provide to smaller firms and focus much too heavily on the transaction costs associated with the massive patent portfolios that the larger industry participants have acquired (the so-called "arms race" build up).[1]

Open source development models work differently. Because open source development proceeds on the premise that no individual or firm will have proprietary control of the software, the firms participating in those development projects might have little need for patents. The cooperative nature of development obviates any need for the actual and implicit cross licensing that provides access to technology throughout the proprietary software sector. The problem, however, is that the open source community does not exist in a vacuum. It exists in a world in which participants in the industry are building up large portfolios of patents, portfolios that pose a serious threat to open source development. Therefore, any thorough analysis of the role of patents in the industry must take account of the effects of the current property rights system on all participants. This essay takes up that issue.

---

[1] Ronald J. Mann, Do Patents Facilitate Financing in the Software Industry?, 83 Texas L. Rev. 961 (2005); Ronald J. Mann & Thomas W. Sager, Patents, Venture Capital, and Software Startups (unpublished 2005 manuscript).

In a nutshell, the problem is that open source developers can (and often do) operate outside of the IP licensing framework that dominates the software industry. Thus, many participants have no patents of their own with which they might protect themselves in IP litigation. At the same time, at least some portions of this community have developed software with a cavalier attitude to the possibility of patent infringement. Those two habits cannot coexist in the long run. If the existing legal framework is not to be abandoned, then the major open source developers must acquire patents of their own or they must gain shelter from the patent portfolios held by those that participate in the proprietary structure.

That raises the question, in turn, whether the potential for high-quality software development through the open source model justifies eradication of software patents for the entire software industry. To put it another way, one potential cost of permitting ready enforcement of software patents is the potential disabling of the open source model. At the same time, a sensible policy analysis must consider the possibilities for the entrepreneurs and small firms struggling to find a foothold in the industry. Because the property rights that patents offer are closely connected with the survival and success of those firms, we must look more closely at the role property rights play in open source before deciding that the need to free open source from the constraints of patents justifies abandoning patents in the industry entirely. Yet it is difficult to analyze that problem definitively in the absence of any objective evidence that would quantify the benefits of open source development or the benefits that the commercial software industry derives from IP.

The problem becomes more difficult when one considers the rapid convergence of commercial and open source licensing models—so that even proprietary companies now often allow access to source code[2] and the prominent open source licenses discussed below regularly permit commercial development of proprietary works derived from the covered products. A complete answer must account for the effects of those licenses on the character of financial investment in open source software. For example, as I discuss below, the restrictions in common open source licenses might tend to tilt the scales in favor of proprietary investments in service firms rather than products firms. If, as seems likely, it is more difficult for startups entering the industry to compete in services sectors than in product sectors, this suggests in turn that the spread of open source software in fact could promote concentration in the software industry.

In this essay, I analyze the role of patent rights in commercialized open source development models—that is, *development* models that are part of *business* models centered on increasing shareholder returns. Section II sets the stage with a brief description of the landscape of the industry and of the licenses on which open source development depends. Section III considers open source as a challenge to the "one-shop" model of proprietary software development, explaining how and why firms in some cases might profit from collaborative development through open source instead of wholly one-shop proprietary development. Finally, Section IV considers the relation between open

[2] For Microsoft's program, see http://www.microsoft.com/resources/sharedsource/default.mspx.

source and the direction and location of innovation in the industry. I write from the perspective that open source will tend to support innovation by larger and better-established firms than wholly proprietary development, which is a model that is at least *relatively* more accessible to startup and younger firms.

A Note on Sources

My account of the industry is based on four sources. The first three are publicly available. First, I have reviewed the existing literature, which includes several serious efforts to analyze the industry.[3] Second, I have read widely in news accounts related to the open source community. Third, to understand how the licenses in the industry actually work, I have studied the texts of the actual licenses with considerable care. Although some scholars have noted the important distinctions in these licenses,[4] the literature generally has failed to consider a link between the specific terms of licenses and the business models that are best suited to using those licenses.

The most important source, however, has been a series of in-depth interviews and site visits at a variety of large and small firms engaged with the open source development model. I have spoken with executives at firms as large as IBM and Microsoft, and also at smaller startup firms and venture capitalists. Because of the sensitive nature of the topics addressed in these interviews, I have adopted a different technique for collecting information than in my previous work. Specifically, contrary to my normal practice, I did not tape or transcribe the interviews, but rather limited myself to detailed contemporaneous notes. I was free to ask any questions I liked. Because of the sensitive nature of the topics at issue here, and because I thought it important to obtain access to frank opinions from executives at large companies, I adopted a much more restrictive framework than has been customary in the interviews for my earlier work. Specifically, the interviews for this project were conducted on the understanding that (I) I would not identify the specific individuals to whom I spoke; (II) I would emphasize that the interviewees expressed their personal views rather than the views of the firms by which they were employed (I emphasize that point here); (III) my notes of the conversations would remain confidential; and (IV) I would not attribute any specific quotations to employees of a particular firm. Because several of the firms were generous enough to give me access to high-ranking executives with decision-making authority related to the subject, I believe that the information from those interviews is uniquely valuable in developing a nuanced understanding of the relation between proprietary and open source

[3] For my purposes, the most noteworthy are Steven Weber, The Success of Open Source (2004); Lawrence Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law (2004); Josh Lerner & Jean Tirole, The Economics of Technology Sharing: Open Source and Beyond (Harvard NOM Research Paper No. 04-35); Josh Lerner & Jean Tirole, The Scope of Open Source Licensing, forthcoming J. L. & Econ. (2005) [hereinafter Lerner & Tirole, Scope of Licensing]; Robert P. Merges, A New Dynamism in the Public Domain, 71 U. Chi. L. Rev. 183 (2004); Philip J. Weiser, The Internet, Innovation, and Intellectual Property Policy, 103 Colum. L. Rev. 534 (2003).

[4] Particularly Rosen, supra note 3, Weber, supra note 3, and Ieuan G. Mahony & Edward J. Naughton, Open Source Software Monetized: Out of the Bazaar and into Big Business, Computer & Internet Lawy., Oct. 2004, at 1.

methods of development. I am confident that I would not have been able to obtain that information by surveys or by formal on-the-record interviews.

Of course, this does raise the possibility of bias, either in reporting the information from my notes or in selecting firms for interviews. For this type of project, perhaps the most that can be said is that I was sensitive to those problems as I reviewed my notes in preparing this manuscript and selected the interview base. In the end, however difficult it might be to replicate this information, I think it is fair to say that it does fall squarely within the relevant methodological tradition in the social sciences. Thus, I think the concerns with replicability go to the weight to be ascribed to the information, rather than its usefulness or validity.[5]

II. The Landscape

A basic understanding of the development and current state of the software industry provides a necessary backdrop to the analytical questions on which this essay focuses. Thus, I start with a broad outline of each model and the core terms of the licenses that shape them.

A. The Proprietary Software Model

1. Formation and Maturation of the Proprietary Software Industry

The software industry formed in the mid-1960s when labor shortages made it increasingly difficult for increasingly complex software to be produced in-house by each computer user as needed.[6] Sales of software products grew rapidly throughout the 1970s, and by the 1980s, the United States had a large and well-developed software industry with more than one thousand firms.[7]

The industry is comprised of two sectors: products and services.[8] The products sector further divides into primarily two markets, one for sales to individuals and the other to businesses (called enterprise software).[9] The enterprise software market, in turn,

---

[5] For general discussion of this sort of qualitative empirical methodology, see Irving Seidman, Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences (2d ed. 1998); Robert K. Yin, Case Study Research: Design and Methods (3rd ed. 2002).

[6] Martin Campbell-Kelly, From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry (2003).

[7] Campbell-Kelly, supra note 6; Michael A. Cusumano, The Business of Software (2004); Vernon W. Ruttan, Technology, Growth and Development: An Induced Innovation Perspective (2001)

[8] Although the general distinction between products and services firms draws on Cusumano, supra note 7, the further breakdown in this paragraph is my own.

[9] A small number of products firms earn revenues in other ways. For example, firms that develop search technology typically rely heavily on advertising revenues.

includes products aimed at software designers and developers,[10] products targeted directly to end users,[11] and products targeted to hardware developers.[12] The services sector is less structured and includes everything from outsourcing the entire IT function, to maintenance contracts, to custom software design, to hosted applications delivered via a web browser. In the last case, the main difference from the license of a prepackaged software product may be that between an upfront license fee and a periodic rental or access fee.

Though firms in the two sectors rely on substantially different business models, the line that separates them is a shifting one. To simplify a complex pattern, it is reasonably accurate to say that products firms are characterized by higher operating margins, higher growth rates, and less stable market shares, whereas services firms have lower operating margins and lower growth rates, but can more readily establish stable market positions.[13] From that perspective, the typical products firm is characterized by high-volume sales of non-customized products that customers can use "off the shelf" with little or no assistance. At the other end of the spectrum are services firms, which generate revenues by helping firms to install, design, and maintain software. In between is a large group of hybrid firms, which generally started by attempting to sell products, but were subsequently forced by market conditions to provide ever-increasing levels of customization, thus degrading their ability to sell high volumes of a pure high-margin product.[14]

As a whole, a remarkable lack of concentration characterizes the software industry. The industry's CR4 ratio is only 39%, and its HHI is less than 600 (where an HHI of 1000 or more qualifies an industry as only moderately concentrated).[15] Census Bureau statistics report more than forty thousand firms in the industry as of 2000.[16]

---

[10] Examples would include web development tools, graphics tools, server software, operating systems, firmware and many others.

[11] These products are likely to be marketed through value-added resellers, channel distributors, system integrators, or independent vendors, and include database programs, office suites, and various vertical industry applications.

[12] Examples here would include the various operating systems and simpler programs developed for integration into the increasingly varied array of electronic devices that rely on computer processing.

[13] Also, as I show in a forthcoming paper with John Allison and Abe Dunn, products firms are more likely to use patents than services firms.

[14] This paragraph summarizes the basic argument of Cusumano, supra note 7.

[15] The industry "CR4" is the "concentration ratio" of, or percentage of total market sales accounted for by, the top 4 software firms. The HHI measures concentration by summing the squares of the individual market shares of all participants. These figures are calculated based on the 2002 software sales figures for the Software 500. They overstate industry concentration to the extent that they ignore software sales by firms outside the Software 500. Conversely, concentration figures would be much higher if the industry were broken down into smaller sectors.

[16] I aggregate data from NAICS 5112 (Software Publishers) and 541511 (Custom Computer Programming Services). The data are available at http://www.census.gov/epcd/ec97/industry/E54151.HTM and http://www.census.gov/epcd/ec97/industry/E5112.HTM (both last visited on Oct. 7, 2003).

Nearly five hundred firms in the industry had a million or more dollars in sales in 2003, even after the contractions in the industry at the turn of the millennium.[17] As I discuss in Section IV, the lack of concentration has considerable implications for the competitive structure of the industry and its openness to innovation.

The lack of concentration is attributable largely to low barriers to entry. Firms typically enter and exit with great frequency.[18] This pattern interacts in a distinct way with the sector designations described above. As the venture capital model that supports most new firms that enter the industry better suits products firms than service firms,[19] products firms are more likely to be the startups receiving financing. New services firms, although not unheard of, are less common and tend to evolve naturally from incumbent or rising product firms adapting to market pressures.

2. Software Licensing under Proprietary Models

An important feature of the evolutionary tension between products firms and services firms is the treatment of source code. Traditionally, hybrid or services firms that sold custom-designed products provided the source code to the user, but with restrictions designed to prevent its further disclosure. However, for prepackaged products, until about 1990, standard license agreements generally did not make the source code available at all. This led to an increasing compatibility problem between software and hardware components because other software developers did not have access to each other's source code.[20]

The rise of the Internet and network computing, both of which have increased the technical complexity of software by orders of magnitude, exacerbated the interoperability problem. This is particularly true for infrastructure and enterprise products, as opposed to end-user applications, which tend to be easier to install). The commoditization of "middleware" products,[21] which made custom software less dominant in that space, also drove the importance of easy compatibility; it is difficult to sell a commodity that does not easily interact with commodity products that provide associated functionality. The complexity underscored the need for transparency in software design, as many

---

[17] http://www.softwaremag.com/sw500 (last visited May 6, 2004).

[18] In 2002, 209 firms received their first round of venture capital financing, receiving a combined total of $872 million (an average of more than $4 million for each firm) during a markedly down year. During 2002, 652 software companies received a total of $4.3 billion (that is, 443 firms received second or subsequent rounds during 2002). Since 1995, 2907 new firms have received venture capital financing. 2003 National Venture Capital Association Yearbook 40. Similarly, the industry's index of substantial operating firms (the Software 500) shows a major turnover each year: there have been about 1100 distinct firms in the Software 500 in the last five years.

[19] Mann & Sager, supra note 1; *see also* Cusumano, supra note 7.

[20] This paragraph and the paragraphs that follow reflect background information derived from conversations with software executives over the last several years.

[21] By middleware, I refer loosely to software that operates as an intermediary between different applications, such as web servers, applications servers, database management systems, and the like.

sophisticated users increasingly began to desire not only a functional software product but also a product that users might be able to understand, replicate, and modify.

Thus, there is a strong market-based need for collaboration in the development of "platform" products, which serve as the backbone of a user's entire system. Although there obviously was competition among firms to own the "platform,"[22] a one-firm platform would present the long-term problem of slowed technological innovation, as that firm's interests naturally would conflict with those of the other firms attempting to provide products and services on the platform.

Theoretically, a workable method for top-down articulation of platform standards or interfaces could have sidestepped the problem. That has not happened, however. One central difficulty is that the industry has not been able to reach a consensus on the relation between patents and standards. Some groups advocate the adoption of standards that will reliably be patent-free, hoping to avoid the possibility that a patentee can tax any substantial portion of standard-based Internet activity. As it happens, however, patented technology has been knowingly adopted into standards in some cases, and there have been several notable incidents where patents were discovered after a standard was implemented.[23]

Others advocate the mandatory licensing of intellectual property rights incorporated into standards. They recognize the difficulty (which should be clear from the discussion below) of establishing a property-preempted zone in which to articulate standards. Even on that issue, however, stakeholders dispute whether the licenses, besides being "reasonable and non-discriminatory" (RAND), must also be royalty free. The most prominent organization, the World Wide Web Consortium (W3C), has generally taken the view that those that participate in a standards process must contribute their patents royalty-free.[24] That approach, however, has the potential to drive patentees from the process, which in turn could deprive the resulting standards of the best technology available. Moreover, if the adopted standard turns out to infringe an essential patent of a departed patentee, then that party could refuse to license its patent entirely or impose unreasonable terms and conditions on those seeking to implement the standard. Thus, many patentees in the industry instead insist that a better approach is to permit a

---

[22] For a theoretical discussion of the economics of that problem, see Douglas Lichtman, Property Rights in Emerging Platform Technologies, 29 J. Legal Stud. 615 (2000). It is a common topic of competitive concern among the executives to whom I have spoken.

[23] Michael G. Cowie & Joseph P. Lavelle, Patents Covering Industry Standards: The Risks to Enforceability Due to Conduct Before Standard-Setting Organizations, 30 AIPLA Q.J. 95 (2002).

[24] For discussion, see Rosen, supra note 3, at 303-11. OASIS recently revised its patent policy to accommodate but not require royalty-free licenses. (http://www.oasis-open.org/who/intellectualproperty.php). Even the W3C policy permits royalties through an "opt-out" provision. *See* http://www.w3.org/Consortium/Patent-Policy-20040205 (section 7).

standard to incorporate patents licensed on a RAND basis even if it is not fully royalty-free.[25]

More recently, cost pressures have given open-source products an important market entry point. To date, that entry point has been in the commercial market, for several reasons. Among other things, those products are attractive because of their ability (discussed in more detail below) to facilitate lower hardware costs by preventing vendor lock-in.[26] It is also certainly important that sophisticated enterprises are more willing to take a risk on a potentially complex installation and integration process.[27] The early dissemination and widespread adoption of Linux and Apache – both free and of demonstrated quality – exemplified these bases for accepting open source products. In contrast, open source has made much more limited inroads in the consumer space. This is true, at least in part, because Microsoft's existing products are much less risky for the typical consumer to install and integrate, yet still offer considerable quality in comparison to existing open-source alternatives.

[25] This paragraph and the preceding one summarize the views of executives in interviews by the author.

[26] See Eric S. Raymond, *The Magic Cauldron* (2000), available at http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/index.html.

[27] A point emphasized in interviews.

3. Cross-Licensing: the Proprietary Equilibrium

As I have explained elsewhere,[28] the widespread use of cross licensing of patented technologies is a key feature of the mature proprietary software development model. The increasing complexity and interdependence of innovation in the industry has made it important for all of the major firms to have access to the intellectual property of the other major firms in the industry. It is likely that many of the most important firms are developing and selling products that at least arguably infringe in some way on patents held by several other major players in the industry. The major firms could test the relative strengths of their portfolio through litigation, but instead have chosen for the most part to enter into a web of cross-licensing agreements. Under those agreements (whether formal and explicit or informal and tacit), most of the large firms generally have access to all of the intellectual property held by most of the other large firms. Thus, those firms for the most part compete against each other based on the strength of their product design and marketing, not on the strength of their IP portfolios.[29]

B. The Open Source Development Model

1. The Rise of Open Source

One development mitigating the tension created by the proprietary licensing model (i.e., the demand/need for source code and resistance to providing it) has been the rise of platform products that are distributed with their source code. Given the practical difficulties of distributing source code with products in a way that ensures that the code will remain confidential, it is perhaps not surprising that the source code is made available not only to the paying users but also to other developers and users at large.

To understand the property rights at issue, it is important to distinguish two stages of an open source project. First, in the contribution stage, dispersed communities of programmers produce lines of code that they contribute to a particular development project. Typically, the copyright in the contributed code rests either with the contributor or with one of several non-profit entities (such as the Free Software Foundation) that acquires the copyright through assignment. Where the copyright is not assigned, the contributor typically licenses the code to the project under the relevant license.[30] Second, in the distribution stage, the software product is distributed under the terms of an open

[28] Mann, supra note 1.

[29] The strength of the IP portfolios is not irrelevant. IBM, for instance, earns a great deal from licensing fees of its portfolio, which plainly is the strongest in the industry. Thus, other firms have an incentive to increase the strength of their portfolios to lower the net sums they must expend on cross-licensing agreements. This effect, however, is relatively indirect, largely because most of the cross-licensing agreements apparently do not involve a transfer of monetary consideration. {The agreements are proprietary, but interviews my suggest (with the notable exception of IBM) the general lack of monetary consideration.}

[30] As discussed below, this can create difficulties when those operating the project later wish to alter the license under which the product is distributed (to "reversion" it, in the typical jargon), because they are likely to need consent from original contributors.

source license. This license restricts the rights of the user in the code (more on that subject in the sections that follow). If the leaders of the project wish for the software to be regarded as "open source," they must select a form of license certified as an open source license by the Open Source Initiative (OSI).[31]

As a method of software production, open source in fact dates to the earliest days of commercial computing, when businesses using IBM computers in the early 1950's collaborated on the task of designing software for their machines.[32] The modern history of open source, however, begins with the birth of UNIX in 1969. Starting from a few months of programming by Ken Thompson at his California home, UNIX developed into a widely used and respected operating system that has become the ultimate source of many of the most successful operating systems in use today.[33]

For purposes of this essay, the most important of the open source projects is GNU, begun by Richard Stallman in 1984 as an effort to create an operating system that would include the benefits of the UNIX operating system but include sufficiently new code to avoid the ownership questions that plagued the distribution of UNIX for decades. GNU became a viable operating system when Linus Torvalds contributed a working kernel to the project in 1994, at which point the software came to be known as GNU/Linux (or confusingly, just Linux). From that point, the Linux operating system has evolved through a rapid collaborative process in which a large, worldwide community of programmers routinely read, redistribute and modify the source code to improve it. It is subject to the General Public License (GPL), one of the earliest, most widely used, and most restrictive of the open source licenses.[34]

The rise of large-scale open source development was facilitated by the birth of the commercial Internet, which has substantially lowered the costs and logistical complexity of transferring information and coordinating programming processes among remote individuals. To that extent, the comparative advantage of the corporate structure as a device for organizing sophisticated research has been challenged by the rise of modern computer-based techniques of collaboration.

As the discussion above suggests, open source holds its greatest promise for platform products. For one thing, the market need is greatest for platform products, because of the importance of a reliable promise that vendor lock-in will not endanger the

---

[31] The Open Source Initiative is a non-profit organization founded in 1998 by Bruce Perens and Eric Raymond. Generally, it supports a broader conception of open source software, more tolerant of commercial interaction, than the Free Software Foundation. For my purposes, the most important of its activities is its promulgation of the Open Source Definition, the generally accepted indicator that a particular license should be regarded as "open source." For details, visit www.opensource.org.

[32] Campbell-Kelly, supra note 6.

[33] Weber, supra note 3, at 20-53.

[34] Weber, supra note 3, at 54-55, 94-109.

survival of products built (or modified) on the software stack above that platform.[35] For another, it is generally more important in ensuring interoperability to have access to the source code of platform products (on which middleware and applications must be stacked) than it is of higher-end applications. For yet another, collaborative development has its highest potential in those areas, where firms specializing in different parts of a value chain have joint incentives to participate in the development of a high-quality product that is broadly accessible. In that context, open source traditionally has been linked to powerful brands, like Linux, Apache, and Perl. Still, some of the modern open source products have moved beyond that niche. The Firefox web browser, for example, is a product gaining recent popularity[36] that is not, at least in its current manifestations, primarily a platform product.[37]

Some of those programs are created almost entirely through the efforts of volunteers,[38] as in the early days of Linux. Even now, it probably still is true that most of the important projects have roots in self-organizing collaborative activity, even if the projects have come to be nurtured and sustained in their maturity by proprietary firms. Still, there has been something of a shift toward proprietary involvement in the initiation of projects. Thus, in recent years proprietary companies have tried – albeit with varying levels of success – to jump-start projects with a release of formerly proprietary code as open source software. Examples here include Netscape's release of its browser source code in 1998 to form the basis of Mozilla,[39] IBM's 2004 release of Cloudscape to the Apache Foundation, and Sun's recent release of the source code for Solaris 10. In still another model, firms fund the development of new projects through a combination of paying employees and sponsoring volunteers to produce products that achieve their goals. Leading examples here would be JBoss and MySQL.

The availability of venture financing – or lack thereof – affects the way in which open source firms enter the market. As discussed in more detail below, it is difficult to obtain financing for a product that will be distributed without charge, for which the

---

[35] Vendor lock-in seems to be a particular concern for government procurement. K.D. Simon, *The Value of Open Standards and Open-Source Software in Government Environments*, 44 IBM SYSTEMS J. 227 (2005).

[36] After releasing several browsers that did not succeed in the market for various reasons, in November 2004, the Mozilla Foundation released Firefox (using second-generation Netscape code). Firefox has been an immediate success. As of August 2005, it is estimated that Firefox has an 8% share of the browser market, compared to 87% for IE and 2% for Safari. Juan Carlos Perez, *Firefox Market Share Slips*, PC WORLD, Aug. 15, 2005, available at www.pcworld.com/news/article/0,aid,122213,00.asp.

[37] To be sure, the rise of "mash-ups" and similar services suggests at least a possibility that the Firefox browser (or some competitor) ultimately will become a major platform for distributed applications. *See* Elinor Mills, *Mapping a Revolution with 'Mashups'*, CNET News.com, Nov. 17, 2005, *available at* http://news.com.com/Mapping+a+revolution+with+mashups/2009-1025_3-5944608.html; Ryan Singel, *Are You Ready for Web 2.0*, WIRED, Oct. 6, 2005, *available at* http://www.wired.com/news/technology/0,1282,69114,00.html.

[38] Sourceforge.net lists tens of thousands of open source projects. However, it seems likely that only a few of those projects have any significant impact on IT. *See* Lerner & Tirole, Scope of Licensing, supra note 3 (analyzing SourceForge data).

[39] The Mozilla Foundation received startup financing from Netscape in 2003.

source code will remain open if the product succeeds, and which (like all software products) may never succeed for technical or market-based reasons. Thus, to the extent it has been available at all, venture financing traditionally has appeared after the open source product is distributed, modified, and already become a market success. For example, when developers at the University of Cambridge developed Xen (software that lets hardware run multiple operating systems) and distributed it openly through two versions, they were then able to form a firm, XenSource, with $6 million of venture backing. That financing was used, in turn, to support work on a third version of the product, the distribution of professional releases tailored for different environments, and product support. The notable point, though, is that the innovative activity preceded the financing. This contrasts starkly with the financing model for firms pursuing proprietary software strategies, where little or no development or deployment is likely to occur before first financing.[40]

Perhaps the most conceptually difficult aspect of the open source development model lies in the way that successful open source projects foster a vibrant and active community of contributors. As many others have recognized, a key part of any such project is designing it in a way that will attract talented and motivated individuals to the project.[41] Generally, the existing literature has focused on how to tap into altruistic motivations for individual participants that may be attracted to participation in a communitarian endeavor.[42] Yet, it is clear that long-time participants in the open-source community experience the success of commercial models founded on open-source community with distaste, feeling that their work is being co-opted by profit-seeking investors and managers.[43] The trends discussed in this essay can only exacerbate that problem.

From my perspective, the need to maintain a community that is attractive to individuals is not a serious problem with the development model. It is simply one feature that affects the way in which projects are designed. For example, it is clear from interviews that the most perceptive proprietary firms that sponsor open source projects will continue to be successful at coordinating their proprietary activities with open source communities that will be interested in participating.[44] For example, IBM's recent donation of the Cloudscape database was governed by an Apache license, at least in part because IBM's working relationship with that community gave IBM confidence not only

---

[40] Mann & Sager, supra note 1.

[41] Yochai Benkler, *Coase's Penguin, or, Linux and the Nature of the Firm*, 112 YALE L.J. 369 (2002); Eric S. Raymond, The Cathedral and the Bazaar (1999).

[42] Justin Pappas Johnson, *Open Source Software: Private Provision of a Public Good*, 11 J. ECONOMICS & MANAGEMENT STRATEGY 637 (2002); Dan M. Kahan, *The Logic of Reciprocity: Trust, Collective Action, and Law*, 102 MICH. L. REV. 71 (2003); Raymond, supra note 26.

[43] E.g., Bruce Perens, *The Emerging Economic Paradigm of Open Source*, First Monday, Oct. 3, 2005, available at http://firstmonday.org/issues/special10_10/.

[44] For a detailed discussion of how to integrate collaborative development into a proprietary firm, see A. Neus & P. Scherf, *Opening Minds: Cultural Change with the Introduction of Open-Source Collaboration Methods*, 44 IBM SYSTEMS J. 215 (2005).

that individuals working in that community would have the skills necessary to make the program successful.[45] It also was important that the relation between that project and other Apache-related projects would make the work sufficiently interesting to attract those individuals into the development community.

2. The Current State of Open Source: Commercialization

The events of the last few years show that the ties between open source communities and large incumbent (proprietary) firms are growing rapidly. Thus, for example, it is plain that a substantial share of the important Linux contributors now has gainful employment either directly for OSDL or for one of its major supporters.[46] Indeed, the location of such a high share of the "important" contributors in such posts is one of the reasons OSDL executives have been optimistic about their ability to obtain consent from enough of those contributors to succeed in reversioning the GPL.[47] Moreover, dual-licensing firms (like MySQL) generally employ directly almost all of those that contribute to their projects. As the interview subjects explain, those firms can simply reject any substantial blocks of code submitted by individuals that are not interested in employment with the company. The increasing ties between proprietary firms and open source projects illustrate how far the open source development model has evolved from the UNIX-hacker days of the 1970s.

It is harder to get a sense of the relation between open source and small firms. Although some of the interviewees suggest that there are a "huge number" of startups building on Linux, it is not clear what to make of that perspective. Using VentureXpert, I found only 131 firms (substantially all of which were founded after 1998) whose business descriptions contain the terms "Linux," "Apache" or "open source." By any standard, that is a small sector of the software startup market. By comparison, for example, more than 200 new software firms received their first financing in 2002 alone.[48] Moreover, few of those 131 firms are actually profiting directly from open source technology. For example, many simply offer heterogeneous (or cross-platform) operating system support, including Linux or Windows,[49] or provide proprietary applications that can be used on

---

[45] For similar accounts of IBM efforts at developing open-source communities, see B. Alpern et al., *The Jikes Research Virtual Machine Project: Building an Open-Source Research Community*, 44 IBM SYSTEMS J. 399 (2005) (Jalapeno); J. Becking et al., *MMBase: An Open-Source Content Management System*, 44 IBM SYSTEMS J. 381 (2005) (MMBase).

[46] *See* Daniel Lyons, *Peace, Love, and Paychecks*, Forbes, Sept. 20, 2004, available at http://www.forbes.com/forbes/2004/0920/180.html/ (discussing corporate sponsorship of key Linux contributors).

[47] *See also* Keith Regan, *Browser Rumors Renewed as Google Hires Firefox Programmer*, E-Commerce Times, Jan. 25, 2005, available at http://www.ecommercetimes.com/story/40015.html (Google hires developer responsible for Firefox browser).

[48] For comparison, in 2002 alone more than 200 new firms received more than $850 million in financing.

[49] I would include here firms like Mission Critical Linux.

either a Windows or Linux platform,[50] or on occasion proprietary applications that can be used only on a Linux platform.[51]

Some of the most interesting startups are not making open source products, but rather are strategically capitalizing on the tension between proprietary and open source development models. Black Duck and Palamida, for example, are two start-up firms that make software designed to assist the commingling of open source and proprietary technology. Several firms sell technology designed to link computers of different operating systems.[52] Open Source Risk Management, for another example, sells legal protection against copyright and patent infringement litigation related to open source products.[53] Finally, some of those firms (like Red Hat, Covalent Technologies, MySQL, JBoss, and formerly SCO Group) are distributors of so-called "professional" open source products, special proprietary or quasi-proprietary versions of traditional open source products.

That is not to say that it is impossible to have a successful venture-backed startup with a purely open-source product. For example, MontaVista Software has been gaining considerable traction in the production of cutting-edge operating systems for embedded devices and cell phones. Currently, it is obtaining license fees for purely open-source operating systems, based almost entirely on its ability to promise speed to the market. With little copyright or patent protection against duplication of its products, that is a difficult route, but it may not be an impossible one.

Another possibility is to start with open source code as the platform on which to build a proprietary product. Several venture capitalists to whom I have spoken suggest that this type of startup is increasingly common.[54] The basic expectation here is that the startups will build proprietary products around the open-source cores, and that the open-source nature of the core will make it easier for the startup to integrate its work with the core. As time goes on, it well may be that this will become an increasingly common method for the development of proprietary software. However, that development is still at an early stage. For now at least, an open source foundation is probably still likely to be an obstacle to sophisticated venture-backed financing.

---

[50] Altiris, Atempo, and PERSIST Technologies.

[51] Aduva, Eazel, Eternal Systems, Qlusters, and Scalix.

[52] Cassatt, Centrify, Steeleye Technology, and Vintela.

[53] There are other firms that are not focusing on open risk management per se, but that are capitalizing on the lack of interoperability between open source and proprietary operating systems. CodeWeavers, for example, offers a software product that facilitates the use of Windows applications on Linux. That product is a "professional" version of the "free software" Wine Project.

[54] See also Martin LaMonica, Open Source, Open Wallet, CNet, Nov. 7, 2005 (discussing VC investments in open-source related startups).

3. Software Products Licenses under Open Source Models

At the center of all of this is the license that governs the use of the code. Before a license can qualify as an open source license, it must have OSI certification. To become certified, the license must meet a set of basic, bare minimum requirements, designed to ensure that software is distributed with its source code and that it be reasonably available without constraint to developers and users that wish to use it or modify it for their own purposes.[55] Again, those requirements are not logically necessary to solve the interoperability and transparency problems discussed above. A proprietary developer could arguably achieve the same ends with an aggressive program of sharing source code with developers and major customers. Indeed, Microsoft's shared source program is designed to address that problem.[56] Yet the absence of any response to those issues throughout the 1990's played a major role in the rise of open source. Moreover, a shared source program cannot solve the concerns about vendor lock-in that motivate many enterprises to choose open rather than proprietary platforms.

Thus, the open source communities' awareness of those issues has led to their establishment of a baseline expectation, embodied in the OSI requirements, that must now be met before any project can take advantage of the formal and informal infrastructure that has arisen to support open source development. Beyond those basic requirements, however, the licenses differ in a number of ways that are important for understanding their effect on commercial development of the licensed software.[57] For the present discussion, the licenses differ most importantly in three ways: (a) the constraints on incorporation of the licensed code in later products; (b) the rules about the

---

[55] As set forth at www.opensource.org, Version 1.9 of the Open Source Definition includes the following requirements: free redistribution must be tolerated; inclusion of source code; the creation and distribution of derivative works must be tolerated; the license cannot discriminate against particular users or fields of endeavor; rights under the license must extend to all users whether or not they have executed a formal license; the license cannot be restricted to use of the program as part of a specific product; the license cannot restrict other software solely because it is distributed with the licensed software; and the license must be technology-neutral.

[56] See http://www.microsoft.com/resources/sharedsource/default.mspx.

[57] An interesting problem that warrants further inquiry is why open source licenses continue to proliferate. *See* Rosen, supra note 3, at 235-38. It would make more sense for a relatively small number of standard forms to begin to dominate, but it continues to be the case that new projects often result in newly developed licenses, like the new Community Development and Distribution License Sun devised for its Solaris contribution. Historically, the classic licenses like the GPL, LGPL, BSD, and MIT licenses dominated significant projects until the late 1990's, but starting with the release of Mozilla in 1998 the number of licenses approved by OSI has increased rapidly. As I write, 58 separate licenses have been approved. This problem has gained increasing attention in recent years, largely because of the increasing difficulty of combining software code written within different licensing domains. The underlying fear is not so much that a particular project (like Linux) will split into separate projects, or fork, as it is that the open source community as a whole will become a number of effectively separate gated communities. Rosen, supra note 3, at 247-53.

contribution of IP rights related to contributed code; and (c) the rules about enforcement of IP rights by users of the software.[58]

The first has traditionally been the major point of differentiation among open source licenses. Here, there is a readily discernible continuum, from fully "reciprocal" licenses (like the GPL) at the one end to "academic" licenses (like the BSD) at the other. The oft-debated § 2(b) of the GPL, for example, provides that its restrictions must apply not only to the original GPL code but also to any "modified work" that includes GPL code unless "identifiable sections" of the modified work "can be reasonably considered independent and separate works in themselves." Thus, the license reflects a concept of reciprocal obligation. If a developer wants to take advantage of the contributions reflected in an existing piece of GPL code, the developer is free to do so, provided that the developer makes a reciprocal contribution of the developer's modifications into the GPL model.[59] The scope of restrictions imposed by that provision is debatable,[60] but it certainly imposes at least some constraint on the ability of a developer to incorporate GPL code into a fully proprietary product.[61]

At the other end of the spectrum, the so-called "academic" licenses like the BSD license impose no similar constraints on distribution, requiring only that distributors include the code and give appropriate credit. The concept of those licenses is that work prepared solely for academic purposes should be freely available to the entire community to use as it sees fit with no strings attached.[62] Thus, for example, Microsoft easily can (and does) include some BSD code in its operating system. Other major licenses have an effect similar to the BSD license, though they state it more explicitly. The Mozilla Public License (MPL), for example, states in § 3.7:

---

[58] This section draws heavily on the terminology and analysis of Rosen, supra note 3.

[59] I do not address here the question whether the licenses are binding as a matter of contract or through rules of property rights. On that point, Peggy Radin has suggested that the property rights argument is quite weak. The absence of robust mechanisms for execution similarly undermines the idea that they operate by creating contractual obligations. Of course, because the right to use the software is likely to depend on the existence of a license, the absence of any contractual obligation will be important only in cases when stopping subsequent use is not an adequate remedy. The main example of this is likely to be in the enforcement of provisions that purport to govern enforcement of patent rights by users of the software.

[60] For contrasting evaluations, compare, *e.g.*, James V. Delong, *The Enigma of Open Source Software* (Version 1.0) (unpublished 2004 manuscript) (a highly expansive interpretation) with Rosen, supra note 3 (a much narrower interpretation).

[61] Typical reciprocity provisions apply only when the work is "distributed." GPL § 2(b). With the rise of application service providers, that leaves a loophole that would permit commercial exploitation of a derivative work without distribution. Accordingly, newer licenses extend the reciprocity provision to include any "external deployment" of the derivative work that makes the work available to users over a computer network. *See, e.g.*, Apple Public Source License § 2.2; Real Networks Public Source License § 1.7; Open Software License § 5. *See also* Rosen, supra note 3, at 193-95.

[62] The concept behind that license resonates strongly with the academic community of motivation and intellectual contribution, as discussed in Rebecca S. Eisenberg, *Proprietary Rights and the Norms of Science in Biotechnology*, 97 YALE L.J. 177 (1987); Rebecca S. Eisenberg, *Academic Freedom and Academic Values in Sponsored Research*, 66 TEXAS L. REV. 1363 (1988).

> You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

{Sun's new Common Development and Distribution License (the CDDL), which governs its contribution of Solaris, includes a substantially identical provision (§ 3.6).} Similarly, the Apache License (Version 2.0) provides in § 4: "You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You [give recipients a copy of the license, include "prominent notices" of your changes, and include appropriate attribution notices]."

The second crucial point of differentiation among the licenses is the coverage of intellectual property rights held by those that contribute to the project. The traditional practice has been to rely on the understanding that any party that contributed to an open source project would grant an implied license that permitted ordinary uses of the resulting software. Licenses like the GPL[63] and the BSD that do not explicitly deal with the problem must rely on that concept.[64] Recent licenses deal with the subject more directly, including specific copyright and patent licenses from all contributors to all users.[65] Indeed, the Apache Software Foundation has developed a separate Apache Contributor License Agreement designed specifically to respond to this problem.[66]

For this discussion, what is most interesting about those licenses is the care with which they limit the patent rights that the contributor grants. For example, § 2.1 of the Mozilla Public License carefully limits the patent grant of the initial developer (Netscape) to cover only patents that are necessary to the use of the Original Code.[67] Thus, if Netscape had *at the time it contributed the Original Code* a patent that was not infringed by the Original Code, but was infringed by a new module added to that code at a later time, nothing in the MPL would require Netscape to license that patent to subsequent users of the code.[68] A slightly different twist comes from IBM's Common Public

---

[63] GPL § 7 does include an odd provision barring redistribution by any party that is prevented by a patent license from tolerating royalty-free distribution. Although that strongly suggests what is obviously expected, it does not rise to the level of an express grant of IP rights by contributors.

[64] As Rosen explains, there are numerous technical problems in relying on implied licenses, such as the question whether the license extends to patents that have not yet been issued at the time of the contribution or to later versions of the open-source project that do not exist at the time of the contribution. Rosen, supra note 3, at 79, 126-127.

[65] *See e.g.*, Apache License §§ 2, 3; CDDL §§ 2.1, 2.2.

[66] *See* Rosen, supra note 3, at 93-94.

[67] It appears that the desire to delimit this grant so carefully was one of the main reasons for the development of the MPL in preference to the then-existing reciprocal license forms. Rosen, supra note 3, at 147-150.

[68] Rosen, supra note 3, at 148-150.

License, which excludes from the patent grant a license to any patent not issued at the time of the contribution, even if an application already was on file.[69]

The final point of differentiation is how the licenses deal with the risk of allegations of patent infringement. On this point, proprietary licenses often indemnify users against patent infringement claims filed by third parties. That is not, however, practical in the open source context. There, the "licensor" of any particular program often is a distributed body of difficult-to-identify contributors.[70] Thus, open source licenses generally impose the infringement risk on licensees.[71] The response to the problem thus is limited to creating incentives of various degrees designed to deter users of the program from instituting patent litigation by the threat of withdrawing further rights to use the open source program.[72] It is difficult to weigh the effect of those provisions. For successful programs that become "mission-critical," it is easy to see that they would have a powerful effect. For less important programs that a user easily could abandon, the provisions would be less effective.

For present purposes, what is most interesting is the great variation in the provisions focusing on third party IP, primarily because it suggests more of a conscious attention to the importance of protecting patent rights than one would expect given the mythology of a patent-free open source movement. For example, § 8 of the MPL provides that a suit claiming that a contributor's version of the software violates a patent will result in a termination of the plaintiff's rights to use that version of the software. Furthermore, a suit against *any* contributor to an MPL project for any other form of patent infringement will lead to a termination of the right to use any contribution of that participant to any MPL product.[73] Perhaps the broadest provision appears in § 12.1(c) of the Apple Public Source License, which terminates "if You * * * commence an action for patent infringement against Apple; provided that Apple did not first commence an action for patent infringement against You."

Those provisions have a fascinating effect, because they generally operate not only to protect the products in question, but as I discuss in more detail below, slowly to bring open source products within the cross-licensing equilibrium that has provided stability to the proprietary wing of the industry for some time. At the same time, those provisions often seem unpalatable to companies with large patent portfolios, because they require them to forgo claims under that portfolio for products unrelated to the open

---

[69] CPL § 2; see Rosen, supra note 3, at 163-166.

[70] That is particularly true for programs governed by licenses like the GPL that do not directly provide for sublicensing, but rather contemplate licenses directly from each contributor to each user.

[71] The closest thing to a warranty of noninfringement is the warranty of "provenance" that appears in many of the modern open source licenses, in which the contributor states it "believes" that its contributions are its original creations and noninfringing. MPL § 3.4(c); *see* Rosen, supra note 3, at 158, 198-201.

[72] *See e.g.*, Apache § 3; MPL § 8.2. As a related matter, licenses also often require contributors to include notice of patent problems of which they might be aware. MPL § 3.4.

[73] Rosen, supra note 3, at 154-56.

source project in which they are participating. This has spurred the drafting of weaker patent defense provisions, such as the one in the current version of § 10 of the Open Software License and the Academic Free License, which terminates a license for the contributed work *only* for a claim against the contributed work.[74] By excluding termination based on the exercise of patent rights against unrelated software, it is thought, the provision makes participation in and use of open source projects more palatable for firms with large patent portfolios.[75]

III. Motivations for the Commercialization of Open Source

A. Open Source as a Viable Business Model

As Section 2 suggests, open source development is aptly viewed as a direct challenge to the traditional "one-shop" model of proprietary software development. The natural question, then, how to assess the relative advantages and disadvantages of the two business models?

The important differences are easily summarized. In the proprietary model, the coordination and direction of research and development are accomplished within the boundaries of a single firm, directed and funded by the management of that firm. The advantages of the model are that of bringing any complex activity within the boundaries of a single firm: the ability of the firm to collect resources from investors and then decide at a central point how best to allocate those resources to maximize the effectiveness of any particular development project. The ability to make rapid responses to new and surprising events, for example, is a strong advantage of the proprietary model.

In the open source model, by contrast, the level of central control is much lower, with development proceeding through relatively decentralized hierarchies.[76] The strength of open source development is its potential to produce products with a higher quality and more innovative character than parallel proprietary products. Although discourse from supporters often reflects a deep-seated, at times almost mystical, conviction that collaborative development is superior to centrally directed development, the argument in fact resonates strongly with the recent literature on open innovation.[77] In that context, advocates have focused on the ability of a collaborative and decentralized development process to produce better solutions more rapidly than a process centralized within a single firm or laboratory. There is also a distinctly populist reveling in the idea that unsupported individuals can produce software of a commercial quality that can compete

---

[74] See also Apache License § 3 (similar provision).

[75] Rosen, supra note 3, at 217-18.

[76] This is not to say that there is not organization. As Weber, supra note 3, explains in detail, there is a great deal of organization of open source development. My point, however, is that control and allocation of resources is decentralized: Linus Torvalds has much less ability than Microsoft's Chief Software Architect to control precisely what Linux projects are handled with what level of urgency and resources.

[77] HENRY WILLIAM CHESBROUGH, OPEN INNOVATION: THE NEW IMPERATIVE FOR CREATING AND PROFITING FROM TECHNOLOGY (2003).

with the output of the world's largest corporations.[78] It is, to be sure, difficult to obtain empirical evidence about quality, and the existing evidence seems ambiguous.[79] The widespread adoption of the commercially successful open-source products, however, offers strong testimony that in some contexts at least the collaborative development model can produce software of high quality and easy interoperability.

Aside from the quality of the software product, there remains the key inquiry of how it can make sense for profit-seeking firms to invest in open source projects if they categorically will be prohibited from obtaining a return on their investment through control of the resulting software. In the proprietary model, property rights make it possible for a firm to internalize the benefits of R&D by excluding third parties from exploiting the results of the research: it is not necessarily easy to make a profit, but it is relatively easy to obtain revenues.

However, the direct importance of property rights to the proprietary software industry can be overstated. Many firms do not directly exploit their patents, and relatively few exploit their patents to collect licensing revenues.[80] One industry executive illustrated that point effectively when he explained that in large patent-sophisticated firms in the software industry there is a split of about 15/85 between patents that are licensed for revenues and those that are used defensively to maintain freedom of action. The analogous ratio in the pharmaceutical industry, he suggested was about 75/25. To the extent that firms do collect licensing revenues,[81] those revenues directly support the R&D that helps the firm to maintain the quality and competitiveness of its technology. Still, the ability to prevent third parties from copying software products (if only through copyright protection) is much more robust in a model with property rights than it is in an open source model, in which the standard OSI requirements – even in the most lenient of licenses – make it impractical to exclude third parties from exploitation of technology created in an open source community. Thus, the open source investor searching for a return must do something other than sell software to customers that wish to use it.

---

[78] A. Boulanger, *Open-Source Versus Proprietary Software: Is One More Reliable and Secure Than the Other*, 44 IBM SYSTEMS J. 239 (2005), provides an interesting (though inconclusive) study of vulnerability and defect rates in open-source and proprietary software.

[79] Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265 (2004). On the one hand, open source proponents can point, among other things, to the low cost of their products (often available for free). At the same time, advocates of proprietary software can point to studies suggesting that the total cost of ownership, including training and maintenance charges, is higher for open source software. As the text suggests, my impression is that the studies as a group are ambiguous, suggesting that one type of software or the other might be cheaper and more effective in one context or another, but that broad general claims of superiority are difficult to sustain.

[80] Mann, supra note 1.

[81] As discussed in Mann, supra note 1, IBM collects literally billions of dollars each year from the licensing of software-related patents. Other incumbent firms, however, have been less successful in generating large revenue streams from those patents. For example, although Microsoft has begun a similar program (also discussed in Mann, supra note 1), it remains to be seen whether it will generate revenues that are sufficiently large to play an important role on Microsoft's income statement.

1. Predatory Motive: the "Kill Microsoft" Approach

In the discussion of possible business models for open source investors, one of the most prominent ideas (suggested, for example, by Rob Merges) is that the model itself cannot be made profitable, but that firms invest in it solely because it decreases the monopoly power of Microsoft.[82] In its simplest form, the idea is that firms are willing to make current expenditures that do not generate a monetary return under current conditions, solely because of the likelihood that they will lessen the ability of Microsoft to extract future monopoly profits in markets in which those firms might eventually participate. To spin it slightly differently, a large customer (like Intel) might like to preserve a competitor to Microsoft simply to minimize the risks of being locked in to a single vendor.

Alternatively, perhaps the expectation is that profits will come from market power in some new market that develops in a more open and competitive way than it would if Microsoft were more powerful. For example, if IBM thinks that it can respond to change and innovation more rapidly than competitors like Microsoft and Sun, then IBM should expect to profit from any development that causes more rapid innovative shifts in the industry.

This explanation is of great concern to Microsoft, where many executives plainly believe that it has some element of truth. However, several software executives to whom I have spoken have emphasized that the most obvious victims of IBM's Linux strategy, to the extent there have been victims, are UNIX competitors like Sun Microsystems, not Microsoft. Sun at one time was a direct competitor with IBM in the market for servers and the software that runs them. The rise of Linux has destabilized Sun's market position as a top-line purveyor of servers and of a state-of-the-art flavor of UNIX (Solaris).[83] Thus, predatory-motive theory seems at best an incomplete story.

2. Traditional Profit Motive: the Value Chain Approach

Although there surely is some truth to it, the "kill Microsoft" explanation understates the extent to which investments in open-source projects are directly profitable – without regard to their effect on Microsoft. Before suggesting that the investments are irrational, it is important to understand how substantial they really are. Executives have estimated that the amount that proprietary companies currently spend on the development of Linux is at least $1 billion a year, much of that coming from a small group of seven large proprietary companies that are major investors in the Open Source Development

---

[82] Merges, supra note 3.

[83] The competition between IBM and Sun is to some degree bound up in their differing open source strategies. IBM of course was one of the earliest of the major proprietary companies to develop a strong open source strategy. Sun's interactions with the movement have been much less harmonious, both because of its decision not to open source Java and because of its willingness to reach a cross-licensing agreement with Microsoft that did not protect Open Office. It remains to be seen whether its decision to open source Solaris in early 2004 will be successful. See also Raymond, supra note 26 (arguing that Sun's license structures have alienated open-source communities).

Laboratory: IBM, HP, Intel, Fujitsu, Red Hat, Novell, and General Motors (collectively the OSDL group).[84]

The most logical explanation for those investments comes from the typical business-school concept of the value chain. The idea is that a successful IT installation necessarily will involve a variety of components, which can be characterized collectively as a value chain (or a software stack). Different companies will have core competencies in different aspects of that chain. One classic strategy is for any company to foster the commoditization of those portions of the stack in which the company does not have a core competency, so that it can earn high(er) returns for those portions of the stack in which it can defeat its competitors.

To use the simplest example, Microsoft and Intel can be seen as developing one successful value chain that involves the sale of highly profitable products paired with the successful commoditization of the personal computer that uses those products. The point is currently easy to see, as the sale of IBM's personal computer division to Lenovo marks the departure of the firm that invented the market, and the increasing domination of a firm the core competency of which is logistics (Dell).

The only departure from the well-recognized strategy described above is to use non-proprietary – "free" – products as part of the value chain instead of commoditized products from other proprietary companies. Conceptually, it is no different from a developer dedicating public streets in a subdivision if that is the way to maximize the total value of the package.[85] Just as all homeowners in an area can benefit by sharing a single public street that runs near all of their homes, the OSDL group benefits by sharing the costs of production of the Linux operating system. For example, one group of executives suggested that maintaining a competitive enterprise software platform currently requires about $500 million of investment each year. If IBM can spend $100

---

[84] This surely understates the total amount of investment. As I have mentioned above, there is some difficult-to-quantify amount of venture-backed investment. There also is a considerable amount of informal investment from proprietary companies that tolerate code writing by their employees or sponsor important open source participants as employees (as when Torvalds worked for some time at Transmeta). MARTIN FINK, THE BUSINESS AND ECONOMICS OF LINUX AND OPEN SOURCE (2003). It also is common for proprietary companies to spin off companies devoted wholly to open source. It is not yet clear, however, how those activities relate to the venture investment activities of major firms. As Benson & Ziedonis show, the investment models for those investments are quite difficult to understand. David Benson & Rosemarie Ziedonis, *Don't Fence Me In: Corporate Venture Capital and the Acquisition of Entrepreneurial Firms* (unpublished 2004 Univ. of Mich. working paper). In this context, I expect that the most difficult to quantify effect is the likelihood that a firm would support an open-source startup that itself might never be profitable if the startup's activities would increase demand for hardware, services, or infrastructure products sold by the sponsor. This surely explains, for example, why Intel Capital is the most prolific investor in the open source-related startups I discuss above. It invested in 14 of the 66 United States firms in the "Computer Software" sector.

[85] *See* Oren Bar-Gill & Gideon Parchomovsky, *The Value of Giving Away Secrets*, 89 VA. L. REV. 1857 (2003) (exploring why an innovator might gain more profit from an innovation if it could foster related innovations through a gift to the public domain of some portion of the innovation).

million per year on Linux and obtain access to such a platform, that is much cheaper than maintaining the platform on its own.[86]

Thus, the individual members of the OSDL group can rationalize their investments in Linux not solely because it might harm Microsoft, and not because of any profits on sales of Linux, but because the development of Linux as a high-quality operating system permits each of them to develop complementary goods and services in their respective core competencies.[87] From this perspective, Linux is not very different from any other vendor providing a software product that can be loaded onto products developed by the firm that contributes to Linux. For those firms, this value chain works better for those companies than the competing Wintel value chain because it is a value chain in which the operating system will not be used to extract profits. Indeed, open source software is optimally suited for this type of arrangement because it is the ultimate commodity: anybody can sell it for free, which makes it quite difficult for a firm to develop a competitive position that allows it to extract profits that would undermine the OSDL strategy. For now at least, this makes cooperation easier because the individual firms more rationally can defer fears that the enterprise will tip in favor of one party's technology that will run the table and drive competitors from the market (as happens so often in proprietary software products markets).

Moreover, when IBM and other members of the OSDL group began making large investments in Linux, Linux already was beginning to make inroads in the server market.[88] If those companies had resolutely stayed outside that field, they risked a disruption in the market – a shift from high-priced servers and proprietary operating systems to commoditized servers with free operating systems that could have driven them from it completely, something that seems about to happen to Sun despite its efforts to participate in the open source community. The textbook response to that situation is to attempt to co-opt the potentially disruptive technology into the business model of the existing firm.[89] The result has not been to stop the disruption. Rather, as suggested above, it has been to focus the disruption on the firms least capable of integrating the new technology into their business models (Sun, if this analysis turns out to be correct).[90] Thus, investment in open source has for those companies (Intel and IBM being potential beneficiaries of Sun's difficulties) been successful as a disruptive strategy.

---

[86] See Raymond, supra note 26 (discussing the benefits of cost-spreading).

[87] Mahony & Naughton, supra note 4; Perens, supra note 43.

[88] For an official IBM account, see P.G. Capek et al., *A History of IBM's Open-Source Involvement and Strategy*, 44 IBM SYSTEMS J. 249 (2005).

[89] Clayton M. Christensen, Jr., The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail (1997). Industry executives emphasized that the rise of Linux does not fit the Christensen model perfectly, largely because Linux entered the market from the top – as a high-quality flexible product, moving from the most demanding users to the least demanding, rather than from the bottom, moving from the least demanding users to the most demanding.

[90] The success of this strategy is particularly noteworthy given the general perception among my interview subjects that Sun's software technology – the Solaris operating system – is the most sophisticated of the Unix-based operating systems.

Therefore, to use an obvious example, IBM is perhaps the most multi-faceted firm in the IT industry. If IBM cannot profit from sales of the Linux operating system and the Apache web server program, there are plenty of ways that it can profit in a value chain that uses those programs. For one thing, it can sell the servers that use those programs. Although IBM has come far from the days when the sale of computer hardware was its only business, it continues to have major hardware lines in the areas where Linux is most commonly used. At the next level, IBM can write proprietary software that can be used on those computers. For example, after IBM failed to write its own successful proprietary web-server program, it surrendered to the dominant Apache program. Having done so, it was able to develop its highly successful WebSphere program, which is designed specifically to run on computers that use Apache. One of the key pieces of that strategy was the ability of IBM to gain market adoption by marketing a program designed for the large community of firms using Apache, something it was unable to do in earlier years when it had tried to bundle similar products with its own proprietary server programs. Finally, and perhaps most profitably, IBM is probably the industry leader at selling the services that are necessary to make the various hardware and software products fit together.[91]

Apple's recent deployment of OS X is another exemplary application of the value-chain approach. There, Apple has deployed a commoditized base of software drawn from the OpenBSD flavor of UNIX, but placed on top of it the sophisticated look and feel of a top-quality graphical user interface – focusing proprietary efforts on Apple's core competency.

B. Open Source as a Market Correction

The most thoughtful assessment of the role of IP in this context is Rob Merges's paper on *A New Dynamism*,[92] which generally portrays open source as a market correction responding to excessive protection of IP. He views the investments that proprietary firms make in open source projects as "property preempting investments" (PPI)—or a form of "anti-property"—designed to protect the commons from enclosure by IP rights held by incumbents (of whom Microsoft is his principal concern). Although

[91] My analysis is not undermined by the examples in Peter Swire's cogent working paper on the security market, *Security Market: Incentives to Disclose for Security and Competitive Reasons* (Oct. 11, 2005) (draft §§ II(B)(3) – (4), discussing this manuscript). Swire suggests that his interviews indicate that proprietary firms are profiting directly from investments in open-source related areas and that my "value-chain" analysis suggests an undue level of indirectness and complication is necessary for proprietary firms to profit in this area. Studying his examples, however, I have the impression that the disagreement is largely semantic. His principal examples – firms that use proprietary code adjoined to open-source code or firms that sell services tailored to open-source code – strike me as precisely the type of business models that I discuss here.

[92] Merges, supra note 3.

that perspective brings a healthy dose of economic analysis to a subject that is often unduly romanticized,[93] I believe that his perspective also is incomplete.

Merges argues that the balance between too many and too few property rights can or will be solved essentially by making PPIs or creation of "anti-property" rights.[94] Generally, he suggests that the investments are designed to make (in my words) an "exclosure" – the opposite of an enclosure – as a "property-free zone"[95] into which later actors cannot force their proprietary claims. That is not, however, a complete answer. To be sure, developers do write lines of code and contribute them to a community under licenses that grant broad use rights to subsequent licensee users. For several reasons, however, that does not have nearly so bucolic an effect as the casual reader of Merges' paper might assume.

The first reason is the simplest one: contributors to open source projects for the most part[96] do not convey their IP rights wholesale to the open source community – they only license them. Therefore, in the case of Linux, there are literally hundreds of contributors that own copyright interests in their contributed code and thereby retain the ability to hinder reversioning of the GPL through exercise of their retained copyright interests.[97] The possibility of conflict is not simply a matter of "FUD"[98]: the analogous reversioning problem for the MPL is at least partially responsible for the birth of Firefox as a substantially new program free from the strictures of the original MPL.

---

[93] *See* Anupam Chandler & Madhavi Sunder, *The Romance of the Public Domain*, 92 CAL. L. REV. 1331 (2004), for a similarly anti-romantic account.

[94] Merges, supra note 3.

[95] I do not want to press the point too far, but the rhetoric of a commons also is inconsistent with the reliance on trademarks, which are critically important to the open source model. *See* Rosen, supra note 3, at 231-32; see also Ingrid Marson, Torvalds Weighs in on Linux Trademark Row, CNet, Aug. 22, 2005 (discussing Linus Torvalds's defense of vigorous action taken on his behalf to enforce the Linux tradename); Ingrid Marson, JBoss Denies Running a Trademark Monopoly, CNet, Oct. 11, 2005 (discussing responses by Marc Fleury to critics of JBoss's trademark enforcement policies). Trademarks have some of the same attributes as other forms of intangible property, such as the creation of network or bandwagon effects. Therefore, even if open source did not depend on patent or copyright protections – a point that I debate in the text – it is still hard to say that property rights are not important in open source.

[96] Although that problem does apply to large portions of the Linux code (apparently because of the expressed wishes of Linus Torvalds), it is by no means universal. Many contributors in fact do convey their rights to entities like the Free Software Foundation or the Apache Foundation, which for my purposes would seem to be a trustee of the "exclosed" commons.

[97] For discussion of the reversioning effort, go to www.gplv3.fsf.org (login required).

[98] According to wikipedia, FUD "is a sales or marketing strategy of disseminating negative but vague or inaccurate information on a competitor's product. The term originated to describe misinformation tactics in the computer software industry and has since been used more broadly. * * * * FUD was first defined by Gene Amdahl after he left IBM to found his own company, Amdahl Corp.: 'FUD is the fear, uncertainty, and doubt that IBM sales people instill in the minds of potential customers who might be considering Amdahl products.'" http://en.wikipedia.org/wiki/FUD

In the event of disagreement over the direction a project should take, ultimately the dispute will be resolved in favor of the person that controls the relevant IP (whether that is copyrights in the source code, control of the trade name, or ownership of important patents).[99] Similarly, reversioning of the GPL would be much easier if everyone in the community could be sure that all of the hundreds of Linux contributors blithely would agree to anything that the organizers of the Free Software Foundation and OSDL submit as an appropriate update of the GPL.[100] But in the end, if there is a dispute over either of those issues, the person with control of the IP will have the final word: it is Torvalds' control of much of the core IP in Linux that gives him so much negotiating power in the struggle to update the GPL.

For still another variation, consider the case of dual-licensing firms like MySQL, where substantially all of the IP rights are localized in the hands of a firm that directly employs contributors to the project, which allows the firm to use a conventional proprietary licensing model to exploit a version of the software that might not differ substantially from the version available under an open-source license.

The second reason why an "exclosure" may not create a fully property-free zone relates to the terms of the open source projects' licenses. As discussed above, it is quite plain, particularly in the area of the modern commercial licenses (the MPL, the CPL, the Apple and Sun licenses, etc.),[101] that licenses are consciously being drafted with considerable technical care to limit the nature of the patent rights a contributor licenses to an open source community. As discussed above, the modern licenses generally do *not* offer a broad grant of all IP rights necessary to permit development of the project to which the contribution has been made. Rather, they are limited to existing patents, or to patents that apply to the project in its current stage, or the like.

The third reason is a simple matter of patent doctrine. Even in situations in which contributors have used licenses or contribution agreements that operate to transfer all of their IP interests, they cannot logically create a property-free "exclosure," because of the possibility that the resulting software product will infringe patent rights held by noncontributors. To be sure, open-source contributions might amount to a sufficiently public use of the code to constitute prior art, and thus would prevent others from obtaining subsequent patent rights that prevented use of the code by the open source community. They would not, however, prevent the assertion of patent rights by persons

---

[99] See, e.g., Stephen Shankland, Open-Source Mambo Project Faces Rift, CNet, Aug. 22, 2005 (discussing dispute among contributors to Mambo).

[100] As I revise this manuscript, the possibility of conflict has become more serious, as Linus Torvalds has announced his dissatisfaction with the early drafts of GPLv3.0. See *Richard Stallman's Radical Approach to Software Patents*, SCI-TECH TODAY, Feb. 8, 2006, *available at* http://www.sci-tech-today.com/news/Stallman-s-Radical-Approach-to-Software/story.xhtml?story_id=0200028E0GNC.

[101] Because the GPL includes *no* explicit patent license from its contributors, it is harder to be precise in making this point about the GPL. I take it as plain, however, that the implied license conveyed by a GPL contribution would be similarly incomplete. *See* Rosen, supra note 3.

who had made similar undisclosed inventions *before* the creation of the open source prior art.

Perhaps the most effective way – albeit an imperfect and costly one – to ensure a zone free of third-party property rights is for the software developer to create its own patent rights to cover the space. For example, Sun claims[102] that it owns all of the patents necessary for the deployment of Solaris. Early and aggressive patenting can make it difficult for independent designers to obtain patent rights that write on to the covered product, though even there the possibility of surprise bombshell patents is a real one in an industry with such a high pace of innovation, where foundational patents easily could issue in 2005 for primeval technology first invented in the distant past of 2001. Thus, many if not all of the large firms in this area continue to collect patents. Although several of those firms have made statements about their plans to enforce or not enforce certain patents against certain potential infringers, none of those firms has made an enforceable commitment to forego their enforcement rights entirely. To the contrary, those patent rights are maintained as part of the elaborate equilibrium of cross-licensing arrangements that I describe above.

To be sure, the early days of 2005 witnessed some notable pledges not to enforce patents by IBM,[103] Sun, and Nokia. Those statements, however, did not extend to contributing the patents to a commons, much less to a property-free public domain. Thus, for example, IBM's pledge was made to developers of open source products and not the public at large.[104] The underlying technology is not, for example, available for the development of proprietary offerings by competing products or services firms (such as Microsoft or Apple, both of which have used UNIX technology in their operating systems). Nor is the grant absolute, because it is not effective against a firm that asserts patent claims against IBM. Similarly, Sun's pledge is limited to patents used in Solaris,[105] so it does little more than the implied (or express) grant of patent rights that

---

[102] This claim seems most implausible, although it has been made quite publicly.

[103] In the case of IBM, the contribution followed a statement that IBM does not intend to assert its patent portfolio against the Linux kernel, unless it is forced to defend itself. That statement is broader, in the sense that it covered the entire portfolio, but it is much less formal, and thus much less likely to create reliably enforceable obligations on the part of IBM as circumstances change in the ever-developing landscape of the industry.

[104] IBM's pledge applies "to any individual, community, or company working on or using software that meets the Open Source Initiative (OSI) definition of open source software now or in the future." The patents (a list of which is available on IBM's web site) cover a broad range of technologies. However, some have criticized the scope of the pledge because many of the patents are thought to be of little use to the open source community. A cursory review of the list reveals that 397 of the 500 patents are in primary IPC G06F (the code typically associated with software). Some of the patents are quite dated: 199 were issued in 2001; 232 were issued in 1997; and 69 were issued in 1993.

[105] Sun's pledge purports to give free access to patents "under the Common Development and Distribution License (CDDL)."

would be included in a license to use Solaris.[106] To be sure, responding to a barrage of criticism of the limited significance of that pledge, Sun has announced that despite the absence of a "fancy pledge" on its website, it has "no intention of suing open source developers."[107] Still, it is not at all clear that Sun has committed that it will not use its patent portfolio to challenge Linux as a competitor to Solaris. To the extent Sun's program rests on the desire to create a Solaris-based value chain that would facilitate the sale of hardware, an attack to destabilize the Linux-based value chain might be a plausible response. The narrowness of the pledges is made clearest by the praise Nokia garnered for the modest step of extending its pledge not only to the existing versions of Linux but also to future ones.[108]

I do not mean to understate the commitment of those firms to the development of collaborative research in those areas. My point is a more fundamental one: that it is not constructive to think of these investments as creating a truly open domain, or in Merges's terms, a "property preempting investment."

Having said that, there can be little doubt that open source strategies are deterring others from *enforcing* their patent rights in some contexts. It works just like the creation of large patent portfolios within individual firms, but is potentially even more powerful. Essentially, using combined patent portfolios to create fences around (at least) some open source technologies, the large firms are just shifting the equilibrium slightly. In substance, they are sending a clear message:

> We mean to protect these technologies as much as – if not more than – we protect our own proprietary products. Although we may not use our patent rights offensively, we will use them to defend our proprietary products *and* the open source technologies that we support.[109]

---

[106] *See* CDDL § 2.1(b). As discussed above, express or implied provisions to that effect are ubiquitous in modern open-source licenses.

[107] Stephen Shankland, *Sun: Patent Use OK Beyond Solaris Project*, Jan. 31, 2005, available at http://news.com.com/Sun+Patent+use+OK+beyond+Solaris+project/2100-7344_3-5557658.html?tag=sas.email

[108] Jim Wagner, *Nokia's Linux Pledge*, Developer, May 26, 2005, available at http://www.internetnews.com/dev-news/article.php/3508146. For an additional anecdote about Computer Associates, compare the laudatory press release discussing Computer Associates' pledge, Chris Preimesberger, CA Patents Made Available to Open-Source Community, eWeek.com, Sept. 7, 2005, with later criticism of the pledge as ineffectual, Matt Whipp, Computer Associates' Patent Donation Is Slammed, PC Pro, Sept. 13, 2005 (reporting complaints about the CA pledge on the grounds that the covered patents had little value).

[109] As I complete this article, several of the major Linux backers have formalized this strategy with the formation of the Open Invention Network. Press reports suggest this entity will provide royalty-free licenses to parties that agree not to assert patent rights against Linux users that have signed similar agreements. Linux Backers Form Patent-Sharing Firm, CNet, Nov. 10, 2005. If the licenses gain broad acceptance, then it could create a shared equilibrium for the patents held by those entities.

IV. The Effect of Commercialized Open Source

A. Effect on Industry Organization and Innovation

If the ultimate effect of the "property-preempting investments" described above really is a shift in the enforcement equilibrium to bring open source programs under the shelter of some of the existing large-firm portfolios, then it is hard to accept the open source phenomenon as a fundamental weakening of the IP system. That is not to say, however, that the rise of open source will not affect innovation in the industry. Recent literature on the relation between IP and industrial organization provides a strong theoretical basis[110] for expecting that the prevailing open source business models will have consequences for the location of innovation.[111] As Tim Wu explains, there is good reason to think that this kind of effect – an effect on the "decision architecture" of an industry – often will be a more important effect of intellectual property rights than a direct effect on competition caused by exploitation of the right to exclude.[112]

I start with the theory articulated by Ashish Arora and his coauthors that a stronger IP system often leads to smaller and more specialized firms.[113] Generally, they reason, strong IP rights encourage investment in specialized firms with a superior ability to innovate, largely because strong IP rights limit the costs of leakage that occurs when the locus of innovation is beyond a firm's boundaries.[114] Conversely, they argue, a weaker IP system makes it more difficult to protect proprietary technology and thus prompts the creation of larger firms and industry consolidation.[115] The effect is particularly salient with technologically intensive inputs, and leads to investments in smaller specialized firms over vertically integrated firms. Research in the chemical

---

[110] In addition to the I/O literature that I discuss below, Petra Moser, *How Do Patent Laws Influence Innovation? Evidence from Nineteenth-Century World Fairs*, 95 AM. ECON. REV. 1215 (2005), reports empirical evidence that stronger IP systems influence the direction of innovation. Although there obviously are numerous potentially overdetermining factors, the recent history of the software industry arguably bears out this point, where we have seen a great deal of innovation at the same time that software patents became easier to obtain.

[111] It is difficult to quantify the effect of stronger or weaker intellectual property systems on levels of innovation. As I explain in Mann, supra note 1, we can say that the levels of innovation in the software industry seem quite high, with R&D intensities greater than in most other industries during the last decade. My point here is simply that the rise of open source is likely to affect the location and dispersion of that innovation.

[112] Tim Wu, Intellectual Property, Innovation, and Decision Architectures, 92 VA. L. REV. (forthcoming 2006).

[113] Ashish Arora & Marco Ceccagnoli, *Patent Protection, Complementary Assets, and Firms' Incentives for Technology Licensing* (unpublished Dec. 16, 2004 manuscript); Ashish Arora & Robert P. Merges, *Property Rights, Firm Boundaries, and R&D Input* (unpublished 2001 manuscript); Ashish Arora & Robert P. Merges, *Specialized Supply Firms, Property Rights and Firm Boundaries*, 13 INDUS. & CORP. CHANGE 451 (2004).

[114] Id.

[115] Id.

industry (by Arora) and semiconductor industry (by Hall and Ziedonis) provides empirical support for that possibility.[116]

The theory that Arora and his co-authors have articulated has obvious applications to the software industry. This is the best example of an industry in which innovation is cumulative, i.e., one in which many firms are attempting to build new products that use the same set of cutting-edge ideas. Thus, a fragmented structure can provide multiple opportunities for solutions to difficult technological problems. This is surely part of the explanation for evidence suggesting that small firms can be more innovative than large firms.

It is also plainly the case that use of property rights to codify the output from research and development makes it *much* easier for firms of differing sizes and research emphases to settle into a cross-licensing equilibrium. Without some form of protection, it would be difficult to force participants in the industry to come to terms regarding how much they should contribute to agreements with their various cross-licensing partners, or to exclude from the equilibrium firms that do not contribute their share of innovation.

Applied to the software industry, it is apparent that as property rights were strengthened in the mid-1990's, the industry became increasingly fragmented. This suggests at least a possibility that the fragmentation has supported a higher rate of innovation than otherwise would have existed. The natural question, then, is whether open source will alter the characteristics that have led to the existing structure. There are good reasons to think – as paradoxical as it might seem – that the rise of open source will support industry *consolidation*, not fragmentation.[117] At bottom, this is because the business models that are most likely to succeed in connection with open source development are business models that work better for larger firms.

The first point relates to credibility. A fundamental distinction between open source and proprietary software is the ambiguity of the sponsor of the program. For proprietary software products, a specific company typically owns, develops, maintains, and supports the program. The purchase of a proprietary software product is, for the most part, a bet that a specific and plainly identifiable company will stand behind the

---

[116] Ashish Arora, Patents, Licensing, and Market Structure in the Chemical Industry (unpublished April 1996 manuscript); 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. ECON. 101 (2001); Rosemarie Ham Ziedonis, *Don't Fence Me in: Fragmented Markets for Technology and the Patent Acquisition Strategies of Firms* (January 2004 draft).

[117] Although it cannot be measured, it may also have implications for the level of innovation. My sense is that corporate participation in the movement—whether or not it succeeds—reflects the fact that the industry has matured to the point that the level of innovation has caught up or is catching up to the needs of users. If innovation is viewed as the commercialization of basic research (perhaps, here, the Internet), then there would be a period of rapid fragmentation and innovation – until the number of possible ways to commercialize the technologies begins to stabilize – followed by a reconsolidation of firms, a lessening of the pace of innovation, and a focus on the efficient delivery of well-defined products and services. At that point, we might expect major breakthroughs to come from academia, government and R&D divisions of large firms until some new "transformative need" is identified or satisfied.

product in a number of important ways. Three of the most important are that the developer will repair flaws in the product promptly; that it will upgrade the product to account for new technological developments; and (most importantly for my analysis) that it will protect users from claims that use of the product infringes the IP rights of third parties.[118]

It may be that proprietary software developers do not often incur ironclad contractual obligations on all of those points. Yet whatever their contracts might say, they certainly have considerable residual legal responsibility for those problems. Moreover, in the reputational marketplace in which software vendors compete for customers, there is a powerful motivation for a software developer to accept responsibility for serious problems related to its software, without regard to the details of its anticipated legal responsibility for those problems.

In contrast, the situation is considerably more complex for open source software. For one thing, the licenses that govern open source software differ from the licenses that govern proprietary software in that the authors of the software are likely to categorically disclaim responsibility for the kinds of problems discussed above.[119] That makes some sense given the nature of the software's development, where specific contributions are made by individuals that cannot expect to use profits from the sale of the software to defray anticipated liabilities that might arise from its distribution and use. Moreover, interviews with industry executives suggest that even the proprietary companies that operate in the open source community almost uniformly disclaim any legal responsibility for problems with the software.

What that means is that any response to users' problems with open source software is likely to come from a reputational constraint rather than some enforceable legal obligation. It is, of course, much more difficult for a business to assess the reliability of a reputational constraint than a legal obligation. Yet it cannot be rational for a business to adopt an open source software platform without satisfying itself that *somebody* will maintain, upgrade, and defend the software.[120]

The most obvious generality must be that a reputational constraint will be more robust for a large and publicly visible firm than it is for a small and emerging one. Detractors of open source software often argue that it is risky for a business to rely on those kinds of constraints for important software purchases. I have no occasion to assess the plausibility of that argument – my point here is simply that a rational business would find it much easier to overcome that concern when the open source software is closely associated with a large and publicly visible firm than when the software is associated

---

[118] Unlike copyright law, which does not control "use" of a copyright work, a patent controls any use of the patented technology.

[119] *E.g.*, Apache License §§ 7-8; BSD ¶2; CDDL § 5; GPL §§ 11-12; MPL § 9. As discussed above, the warranty of provenance that appears in a few of the more recently drafted open source licenses is the most important qualification to the assertion in the text.

[120] Fink, supra note 84; Weiser, supra note 3.

with a smaller or younger firm. It is no accident that open source's commercial success has risen rapidly since IBM's public embrace of Apache and Linux at the beginning of this decade. So, for example, assuming products of similar quality, it seems plain that even a relatively small publicly traded firm like Pervasive would have an advantage in finding customers for an open-source database product over smaller startup firms purveying similar products (like Green Plum).

The second point relates to the distinction between products and services firms. As discussed above, the open source model leans ineluctably toward services firms, particularly when the underlying open source project is governed by the GPL. Thus, the open source model tends to lead to a population of services firms customizing and integrating software and hardware for enterprise customers, while the proprietary model more often leads to a set of products firms selling off-the-shelf products that may trade off specialized functionality for ease of installation and maintenance. This is of course a generalization – there are open-source products firms (like MontaVista and the startups discussed above) and important proprietary services firms. But the constraints of the business model do press open source firms toward the services end of the spectrum more forcefully than they do proprietary firms.

To the extent that this theory is true, the open source model should in turn support larger firms because larger firms have a comparative advantage in the service sectors of the software industry. A few overlapping reasons give rise to this comparative advantage. First, the VC startup model works much better for products firms than it does for services firms;[121] thus, there will be relatively few startup services firms.[122] Second, there is good reason to think that the property rights granted by patents will be uniquely valuable to firms attempting to progress successfully through the venture-backed stage.[123]

The comparative advantage continues throughout the business cycle. Just as the products model is better suited to the venture-backed financing common for startups, large established firms will have an advantage in service sectors.[124] For one thing, large incumbent firms are simply going to be better at the integrative services model epitomized by IBM. The "not flashy, just fully informed" business is nearly always going to be the large established firm, not the destructive innovator. For another, as I heard repeatedly in interviews, there are considerable economies of scale in providing the

---

[121] Cusumano, supra note 7.

[122] Mann, supra note 1; Mann & Sager, supra note 1.

[123] Mann, supra note 1; Mann & Sager, supra note 1. That relation is affected by the point in my forthcoming paper with John Allison and Abe Dunn, presenting empirical evidence that patents are more commonly used by products firms than by services firms.

[124] Subscription sales carry high margins in the range of 88-89%. Services sales carry much lower margins, in the range of 43-44%. In addition, although services revenues have remained relatively flat, subscription revenues skyrocketed in 2004 when Red Hat sold approximately 169,500 subscriptions to RHEL products compared to 36,500 in the previous year. Novell seems to be entering a similar phase, with much higher margins on software licenses than on services (90% versus 50% in 2002), but steeper declines in licensing sales. It remains to be seen whether Novell's accelerating shift to a Linux platform can stem the decline.

kind of 24/7 quick-response service that large corporations expect from their software providers: it is much harder for a startup with three customers to support that kind of infrastructure than a larger company with dozens (or hundreds) of customers to support.

Red Hat is perhaps the best example of this. After raising $13 million from venture capitalists and strategic investors in 1998 and early 1999, Red Hat raised $84 million in an August 1999 IPO. However, even Red Hat was unable to achieve profitability using a traditional services model coupled with a pure open-source product. Red Hat never turned a profit until its decision in 2002 to split its product line between the slow-changing Red Hat Enterprise Linux (RHEL), which comes with certifications, long-term guarantees for support and bug fixes, and a mandatory per-computer price tag; and the fast-changing Fedora, which is free, uncertified, relatively unsupported, and packed with the latest upgrades. Selling annual subscriptions to RHEL helped push Red Hat into profitability for the first time in 2004.[125]

More generally, we could say that a property rights system favors new entrants because large firms can use other tools related to their market power to continue to grow (leveraging products against other products, leveraging services against products, marketing advantages, etc.). Small firms have nowhere to turn except property rights. More pointedly, it is much easier for a small startup to pursue an idea to the point of having a solid patent or set of patents sufficient to protect the idea from competitors than it is to develop the kind of brand identification and market power that would make it a strong competitor against the large incumbent firms in the industry. In substance (as Figure One suggests), this is a basic distinction in the types of appropriation mechanisms that are useful for different types of firms.

FIGURE ONE: APPROPRIATION MECHANISMS

SMALLER FIRMS	LARGER FIRMS
First-Mover Advantage	Market Power
Patents	Brand Identification
	Leveraging Value Chains

From this perspective, what open source does, in the sectors where it succeeds, is to remove from the market firms that are developing discrete products from which they wish to get revenues.[126] So we see that a fuller vision of open source reveals a great interconnection, if not outright dependence, on proprietary property rights, and the

[125] See Dwight Johnson, Venture Capital Invested in Red Hat, Linux Journal, Dec. 01, 1998, available at http://www.linuxjournal.com/article/3171; Stephen Shankland, Red Hat Shares Triple in IPO, CNET, Aug. 11, 1999; Red Hat 10-K (May 16 2005)

[126] The point can be pressed too far. As discussed above, there is some reason to think that open-source software can be a useful input for proprietary products startups, by decreasing the costs of completing a marketable product and helping firms to focus their development expenditures on the portions of their product that are uniquely differentiating. Those firms, of course, depend directly on the property rights in the products that they sell.

potential that it might support a substantial shift in the dispersion of innovation in the industry. Thus, it is open source, and not patents, that pose the biggest threat to the "polyarchic" decision structure under which the software industry has flourished for the last decade.[127]

Taking the point further, it is possible that open source might disrupt not only the ability of small firms to grow and innovate, it also might disrupt the relatively stable equilibrium under which the industry has grown in the last several years.[128] The legal dispute over Linux plainly has the potential to disrupt the distribution of Linux-related products and services.[129] Open source still has many unanswered questions: What happens if one of the many individual contributors to an open source program provides even a few lines of code that contain the trade secrets of another firm? Or that infringe another firm's copyright or patent? Would removal of the infringing lines be an adequate response? Or would a court enjoin distribution of the entire program? Or require the payment of substantial damages by any and all of the many users of the program?

Thus, we see the industry at a turning point. The rapid growth of property rights in the industry over the last decade has had a relatively benign effect so far, largely because of the relatively stable equilibrium that has prevailed until now.[130] But can the open source business models discussed above grow to maturity without collapsing that equilibrium? Will one method of development or the other prevail so completely as to dominate the industry?

Some of those questions are directly at issue in the SCO litigation. Others are implicit. For example, the case directly raises the possibility that a court might hold the GPL unenforceable.[131] Does it create a binding contract?[132] Will it be enforced as written? Will anyone who distributes open source software be forever barred from enforcing property rights? And, will large patentees such as IBM use their patents as a

---

[127] I take the term from Wu, supra note 112.

[128] Mann, supra note 1; Mann & Sager, supra note 1.

[129] SCO's lawsuit contends that IBM obtained information concerning the UNIX source code and derivative works from SCO and inappropriately used and distributed that information in connection with its efforts to promote Linux. IBM has responded vigorously, claiming that SCO does not have the right to assert claims based on UNIX ownership, that SCO has breached the GPL, and that SCO has infringed certain patents owned by IBM. Although early news reports predicted that the lawsuit would harm Linux, others have now claimed that the suit actually has helped Linux by accelerating its popularity and legal foundation. Stuart Cohen, *How SCO's Threats Rallied Linux*, BUSINESS WEEK ONLINE, Feb. 7, 2005, available at http://www.businessweek.com/technology/content/feb2005/tc2005027_4780.htm. For a discussion of the murky history of IP rights in UNIX as resolved in the AT&T/Berkeley litigation, see Weber, supra note 3, at 49-52.

[130] Mann, supra note 1; Mann & Sager, supra note 1.

[131] For a recent decision holding it enforceable, see In re Welte, No. 21 O 6123/04 (Dist. Ct. of Munich May 19, 2004), available at http://www.utexas.edu/law/faculty/ecommerce/2nd/assignments/25/Munich%20GPL%20Decision.pdf.

[132] *See* Rosen, supra note 3, for a cogent discussion of that issue.

club to protect just Linux or will those protections extend to other open source programs? The way the industry is responding to those unsettled questions is fascinating; the answers will likely reveal the direction of the industry in the years to come.

Another important question is the significance of the various development communities arising in the open source space. If large firms take over much of the open source software, what will happen to those communities? Richard Epstein, for example, pointedly questions whether loose networks of affiliated firms can survive without the corporate governance structures that support the single-firm models.[133] His question, applied to open source communities, raises two related points.

The first point relates to the licenses. In the current environment, at least, there is some reason to be concerned about the stability of the existing licenses, and in particular, the stability of communities built on reciprocal licenses like the GPL. As discussed above, reciprocal licenses like the GPL[134] impose greater restrictions on the ability of proprietary firms to integrate their products with open source code than academic licenses or many of the more recently developed commercial licenses. For the outsider, it is difficult to assess the significance of that point. Thus, it is clear from the interviews I conducted that sophisticated developers can use techniques to write programs that are adequately functional and yet technically maintain the separation from the operating system's kernel that is necessary to avoid "infection" by the GPL license. What is not clear, however, is how much effort is required for engineers of less than complete sophistication to invent or master those techniques. The interviews leave me with the strong impression that this is a serious problem for all but the most elite organizations. This suggests a minor point of some irony – that increasing (or decreasing) use of the GPL interacts with a relative advantage that larger companies have in working on the fringes of GPL projects.

The second point is that the type of license will likely have some effect on the type of software firm that effectively can use the project. For example, it is widely recognized that a more lenient license permits *more* third-party development.[135] Previous scholars have not focused, however, on the likely effects that differing licenses have on the types of third-party development. A strong reciprocal license (like the GPL) works fine for a services firm. A firm that profits from services should be relatively agnostic about the commoditization of the software that their customers buy.[136] Indeed, they should prefer Linux to the extent that Linux is more of a hybrid product than competing

---

[133] Richard A. Epstein, *Why Open Source Is Unsustainable*, Financial Times, Oct. 21, 2004, *available at* http://news.ft.com/cms/s/78d9812a-2386-11d9-aee5-00000e2511c8,ft_acl=,s01=1.html (last visited January 28, 2005).

[134] It is clear that a substantial part of the concern about the GPL relates to its ambiguity rather than its substantive terms standing alone. *E.g.*, Robert W. Gomulkiewicz, *De-Bugging Open Source Software Licensing*, 64 U. Pitt. L. Rev. 75, 83-92 (2002). Whether that can be solved by reversioning is yet another difficult question for the communities that rely on that license.

[135] Weber, supra note 3.

[136] Cusumano, supra note 7.

proprietary products – thereby shifting a greater share of the cost of ownership to services related to installation and integration. Thus, for example, firms like Cygnus Solutions, Linuxcare, Turbolinux, and Red Hat have developed business models for selling consulting and services related to Linux software. VA Linux, like IBM, sells both consulting services and servers.

In contrast to services firms, it is much easier for a products firm to operate in an environment with a less restrictive license (like Apache or a BSD-space). Thus, small firms like Covalent and Gluecode develop proprietary software that is designed to operate with Apache software. More importantly, some of the most highly visible and successful proprietary software products have been built on top of code covered by such licenses. The most prominent example of course is IBM's WebSphere program, discussed above, which is built on and interacts directly with the Apache Web server. More recently, Apple's widely acclaimed new operating system – OS X – rests on top of a BSD-licensed operating system (FreeBSD 3.2): Apple has layered its popular graphic user interface (GUI) onto the UNIX style open source operating system. Executives with whom I discussed that subject pointed to this as one of the most perceptive executions of a core competency strategy: Apple maintains control of the GUI that gives its products so much verve in the marketplace, but takes advantage of the commoditized operating system available from the open source community.

As the commercialization of the open source model proceeds, the pressure placed on the GPL will necessarily increase. If it turns out that it is important in the marketplace for there to be proprietary products more closely related to the Linux kernel than the GPL permits, the open source movement will confront a contracting crisis in which the software must suffer in functionality unless the GPL can be revised to accommodate those concerns. As Epstein notes, that is the scenario that poses the greatest challenge to the decentralized development model: the lack of a single control point makes a substantial shift in direction more difficult than it is for a proprietary firm.[137] The Linux community is aware of the problem, as it enters a period of "reversioning" designed to promulgate a new version of the GPL that would cover subsequent distributions of Linux.[138] The inability of Mozilla to pass through reversioning successfully makes this process an important one for the open source communities: can they develop the institutional structures to modify the contracts successfully (as they hope) or will they be forced periodically to start over (as Firefox has done for the most part in the browser market)? Long-term commercial success probably depends on the ability of the proponents of Linux to persuade users that they will *not* need to start over simply to resolve licensing problems.

Beyond those short term problems lurk longer-term problems. Academics commonly have noted the apparent incongruity in the large-scale voluntary efforts of open source contributors to develop commercially valuable software.[139] A sophisticated

[137] Epstein, supra note 133.

[138] See supra note 97.

[139] Benkler, supra note 41.

literature offers a variety of reasons why individuals might make such contributions.[140] As commercialization proceeds, however, and firms justify their contributions by reference to the kinds of value-chain motivations discussed above, there is a risk that long-term shifts in market structure will cause those motivations to dissipate. For example, it makes sense in current conditions for all of the various players in the OSDL group to make substantial contributions of personnel, technology, and resources to Linux development. However, as the market shifts and different firms gain dominance, there always is the possibility that the community capable of profiting from Linux-related products may contract (just as it might grow). Thus, Dell recently withdrew from OSDL, apparently (according to comments in my interviews) concluding that the markets in which it could exploit its core capabilities were not sufficiently related to Linux products to justify continued contributions.

This is, at its core, a classic free-rider problem: if some of the contributors are profiting from value-chain investments in Linux and others are not, why shouldn't those who are not profiting (or are profiting less) withdraw from the process (or diminish their contributions). Once that process begins, it might rapidly reach a tipping point where commercial contributions became limited to a relatively small number of firms.

Broader concerns relate to the continuing efforts of firms to use the contracts that organize their communities to design novel and specialized types of communities – just as the real-estate developer uses covenants and restrictions to erect a particular set of property rights tailor-made to a particular subdivision. Existing practices suggest a spectrum ranging from complete enclosure in a single firm to open access to all.[141]

The first step along this spectrum is the proprietary development system, exemplified by Apple's desktop computers, which traditionally have used an operating system with a completely proprietary interface that allows Apple to control not only the basic products, but also the applications and utilities that interact with those products. The vigor of that model is consistent with the 2004 dispute between Apple and Real Networks over Real's efforts to develop software compatible with Apple's highly successful iPod. That development model has allowed Apple to develop products that many users regard as the ultimate in functionality and ease of integration, though it of course has had drawbacks in limiting the size of the development community that produces applications for those products.[142]

The second step is proprietary development with an open interface. Microsoft's products typically have joined closely guarded proprietary code with relatively easy access to interfaces, allowing third parties to develop compatible products. That model has given Microsoft strong market power in the market for operating systems and office applications for desktop computers, both because of Microsoft's enormous investments in

---

[140] *See, e.g.*, Kahan, supra note 42.

[141] *See* Saul Levmore, *Property's Uneasy Path and Expanding Future*, 70 U. CHI. L. REV. 181 (2003) (suggesting that this is a foundational distinction).

[142] Lichtman, supra note 22.

continuing development of its software and because of the substantial community of third-party developers whose products have extended the functionality of Microsoft's software.

A third step is a gated community. A good example from my interviews appears in the context of semiconductor development. In that industry, there are two substantial competing camps of research and development. Intel of course leads one of them. The other is a consortium of researchers from IBM, Advanced Micro Devices, and others. All members of the consortium contribute funds and personnel, and gain access to a pooled set of IP, but the IP is not available to nonmembers (which is to say, Intel). Industry executives praised the success of this development model, which has produced technology commensurate with Intel's technology at a much lower cost. Although this type of community is formally proprietary, the practical import is quite similar to the modern commercial open source community. As discussed above, the patent licenses typically offered by open source contributors are so carefully restricted as to limit the freedom of outsiders to use the technology or take it in directions contrary to the wishes of the sponsors and major contributors.

The final step on the spectrum is the wholly open community, characterized (at least in theory)[143] by the Linux community governed by the GPL. The business case for this community is openly one of collaborative development. As suggested above, that model offers benefits both from the cost savings of collaborative development and from the technological gains of collaborative rather than one-shop development.[144]

It is clear that major market players are constantly developing new consortia, on a case-by-case basis, which reflect different structures for investment in and access to technology, tailored to different user markets. At the most simplistic level, development at the proprietary end of the spectrum is more suited for products aimed at individuals, such as desktop applications, where ease of installation and network effects are integral to market penetration. At the opposite end, wholly open solutions often are attractive for applications targeted at sophisticated users in enterprise settings where factors like total cost of ownership and specialized integration with other systems are more important than ease of installation. As those structures become increasingly specialized and numerous, will their informality be able to survive? As the discussion above suggests, any number of events could destabilize those communities – a tipping toward the technology of a particular partner, or an incendiary assertion of property rights by somebody outside the community.

B. Effect on Intellectual Property Rights

---

[143] Given the implied nature of the GPL patent license, there is as I have explained a great deal of doubt as to exactly how free users are to take and modify Linux. I take it as a practical reality, however, that the contributors with patent rights in that area are unlikely to enforce them against users or modifiers of Linux.

[144] Chesbrough, supra note 77.

The most pointed question is the importance of property rights in the years to come. There are some contractual efforts to limit the importance of those claims. For example, firms like Microsoft and HP have begun to indemnify their users from potential infringement claims.[145] Others have begun to offer insurance policies.[146] Similarly, one of the distinguishing features of the subscription models that are developing for open-source products is contractual provisions that protect customers from IP claims related to the open-source product.[147] As discussed above, still others are promising *not* to enforce existing property rights, or (at least implicitly) promising *to* enforce existing property rights to harm those who threaten the movement.

Still, most agree that it is necessary to acquire more patents to play the game. Hence, it is relevant that the major corporate members of the OSDL group continue to make heavy investments in patented technology: IBM and HP, for example have both recently been obtaining new software patents at a pace of more than 1000 patents/year. Similarly, press reports suggest that pure open source firms increasingly are attempting to acquire their own patents, primarily to protect themselves against the threat of litigation.[148] There may have been a time when the open source community was dominated by a political motivation not to obtain software patents, but that time is fading rapidly into the past. In addition, it is worth noting that most software patents are issued to firms outside the industry.[149] There is no reason to think that an increase – or decrease – in collaborative development in the software industry will have a substantial effect on the propensity of firms outside the industry to obtain patents.

Finally, and most fundamentally, it is not clear that the models of the corporate participants (like the OSDL group) would work without the internalization of R&D rights that patents facilitate. If much of the participation of proprietary firms in open source development is motivated by "value-chain" returns – i.e., firms accepting (or fostering) commoditization at one point of a value chain at the same time they attempt to stake out a profitable point of competency elsewhere on the set of products and services their

---

[145] Microsoft offers a broad protection against infringement claims based on its products. Stacey Higginbotham, How Open? That's the Big Patent Question, The Deal.Com, Sept. 25, 2005. Hewlett-Packard's indemnification plan protects against SCO attacks. Similarly, Red Hat offers a warranty to replace any infringing code. Novell offers some legal indemnification against copyright infringement claims brought against Linux server software customers, and has threatened to use its patent portfolio to defend patent claims brought against itself or one of its customers involving its open source products. Stephen Shankland, *Novell Offers Legal Protection for Linux*, Jan. 13, 2004, available at http://news.com.com/Novell+offers+legal+protection+for+Linux/2100-7344_3-5139632.html?tag=nl.

[146] This is the business model of Open Source Risk Management. As I write, that company is in advanced negotiations under which Lloyds would underwrite offer insurance against claims of copyright infringement through the use of Linux. Ingrid Marson, *Lloyd's May Offer Open-Source Indemnity*, Aug. 15, 2005, available at http://news.com.com/Lloyds+may+offer+open-source+indemnity/2100-7344_3-5833077.html?tag=sas.email.

[147] Examples here from my interviews would be firms like MySQL, Pervasive, and Red Hat.

[148] E.g., Joe Brockmeier, Red Hat Acquiring Netscape Enterprise Solutions Software, lwn.net, Oct. 6, 2004

[149] James Bessen & Robert Hunt, An Empirical Look at Software Patents (Mar. 2004) (unpublished manuscript).

customers use – then patents presumably will be just as important (if not more so) in the remaining core areas in which those firms are attempting to differentiate themselves. Thus, IBM's willingness to refrain from enforcing some of its patents against open source developers does not carry with it a willingness to forgo the use of IP to protect its proprietary products like WebSphere. It may be that patents are less useful for the services portion of IBM's business than they are for its products or hardware sectors, but there is little reason to believe that any of those lines of activity would benefit from the removal of patent protection. Nor has Sun granted free access to its portfolio – it has granted only the patent rights necessary for the use of the specific program that it has contributed to the open source community.

The core issue, I think, is the question of the significance of the threat of patent infringement litigation. The mere threat has had numerous effects, ranging from the contractual assurances discussed above, to some difficult-to-gauge disruption of Linux adoption. There also have been a number of events suggesting the possibility that large firms not participating in open source value chains might exploit their portfolio against those that are participating in those value chains.[150] It is not clear, however, that the enforcement risk in fact is substantial. First and most important, to date the risk remains only hypothetical: there has not yet been a patent infringement suit filed challenging the use or development of Linux or any other open source program of which I am aware.

Second, the risk of litigation is easily overstated. The basic concern is as follows: (I) that there are literally thousands of existing software patents that cover the subjects of open source software programs – perhaps 10,000 patents that cover operating systems alone; (II) that open source communities as a matter of disdain for the practices of proprietary firms do not sully themselves with obtaining patents; (III) that infringement by open source products is more detectable than infringement by proprietary products, because the source code will more directly display the algorithm of the software; and thus (IV) that users of open source software are at a substantial risk of liability for patent infringement.[151] A prominent study by Open Source Risk Management (OSRM), for example, concluded that the Linux kernel infringes 283 currently issued patents.[152] The OSRM business model, however, is predicated on the seriousness of that risk, which makes it difficult to weigh the credibility of its study.

Two points make me doubt the significance of the patent infringement threat. The first and most important is the limited usefulness of patents as a tool for appropriating innovations in software. As I explain in detail in a related paper, most patents on software innovations are not sufficiently robust to prevent competitors from developing non-infringing programs that include the functionality of the innovation represented by

---

[150] The most salient event here probably is the controversy over Sun's willingness to enter into a cross-licensing agreement with Microsoft that extends protection from Microsoft's portfolio to Sun's proprietary products but not to OpenOffice.

[151] Zittrain, supra note 79.

[152] Sean Michael Kerner, *Linux's Patent Risk* (Aug 2, 2004), available at www.Internetnews.com (last visited Nov. 22, 2004).

the patent.[153] This is true for a variety of reasons, the most important of which is the pattern of software innovation that provides multiple paths to most design problems.

The second, related, point is the fact that at least some of the largest firms in the industry have patent-reviewing practices that undermine the possibility and extent of interference from issued patents. Executives at more than one of the firms with whom I spoke indicated that they have routine programs that monitor patents as they are issued, watching for patents that might write onto products of the firm. It is common for the lawyers in these programs to discover such patents, and for the firms to respond promptly to alleviate the problem. Depending on the seriousness of the problem, a range of obvious responses appear: ignore the patent, on the premise that it is either invalid or does not extend to the product in question; rewrite the software to avoid the patent; obtain a license from the patentee; or acquire the patentee (or the patent).

It seems clear that all of these responses (and more complex combinations of them) are common in the industry. It also is clear that the OSDL group engages in similar activity with respect to commercially significant open source programs like Linux and Apache. Collectively, those two points undermine step (IV) of the syllogism above, because they suggest that however many patents there might be that affect commercially important open source programs, the likelihood of a serious problem of infringement is relatively slight.

Still, it is at least plausible that there are patents that might pose problems for open source programs.[154] One highly informed executive, for example, suggested that there are about 200 crucial patents, access to which is necessary to distribute a modern operating system. Although that estimate seems quite high, even a much lower estimate would suggest a serious potential for infringement by open source programs that do not have access to patented technology. Here, however, is where step (II) of the syllogism breaks down. As discussed above, the "disdain" argument no longer has any descriptive force for the modern open source development community. Whatever might be true for the "hard-core" portions of the community associated with the Free Software Foundation, nothing could be further from the truth for players like the OSDL group that are fostering the commercially important open source programs. There is every reason to believe that those firms will make their patents available to the extent necessary to protect users of open source software. Indeed, one executive at an OSDL firm suggested that a relatively common response to the issuance of a third-party patent that affects Linux is for a member of the OSDL group to grant formal access to a patent necessary to write around the third-party patent.

---

[153] Mann, supra note 1.

[154] Perhaps the most significant event to date on that front is Kodak's recent successful lawsuit against Sun claiming that Sun's Java technology infringed aged Kodak patents inherited from Wang Laboratories. Industry observers worry that the basic operations at issue in the litigation are used in a variety of existing open source products, which thus could be vulnerable to similar challenges by Kodak. Given Kodak's relatively poor market position in its major existing business models, Kodak has not that much more to lose by aggressive litigation here than SCO does.

With that information in mind, we might think that the likelihood of a risk of infringement is about as serious for the commercially important proprietary products (from firms like Microsoft and Adobe) as it is for open source products. In either case, the problem will be serious only if the two camps go into an open war enforcing their patents against each other – an outcome that seems most unlikely under current conditions, for reasons explained in a related paper[155] – or if a party such as a troll that is outside the existing equilibrium holds the patent. In sum, it certainly is plausible that a troll could obtain a "nuclear bomb" patent that would write onto major commercial software platforms. It is hard to say, clear, however, that Linux is categorically more vulnerable to such an attack than Windows, if only because the OSDL group collectively has many more patents with which to justify its activities than any single developer of proprietary products (even Microsoft). A single shared pool among all in the industry might resist such an attack more readily than the silos of patents that currently exist, but it is not obvious that one or the other silo is less capable of protecting itself.

V. Conclusion

Some academics see the open source movement as a victim of an excessive IP system and fear that it cannot co-exist with the commercial development model, which depends on increasingly large patent portfolios. Others see it as the best antidote for a broken IP system and hope that it will force software firms to gravitate towards business models that do not rely on IP protections, even if those models provide lower returns. Still others see the movement as a case study on the unsuitability of traditional development models that depend on appropriating the returns to R&D through IP investments and predict the abandonment of IP-centric development models.

This essay fleshes out those ideas and tests their limits. The foundational claim of this paper is that the open source model is largely *consistent* with current economic theories about optimal ways of leveraging R&D to serve the distinct needs of different end-user markets. I argue that commercial participants form collaborative development communities that mirror the more typical firm-based development processes that depend directly on off-the-shelf IP rules. They do this when it is more efficient to invest in inter-firm innovative activities, and they use traditional appropriation mechanisms when intra-firm activities make more sense.

It is difficult to assess whether either model would be more successful without the influence of the other. Given the lower returns experienced by some of the commercial participants, there is some reason to believe that firms are being drawn to open source development as a second-best outcome: as it becomes increasingly difficult to maintain competitive differentiation with a traditional development structure, open source offers a promising alternative tactic. The ongoing strategic repositioning renders the structure of the industry far too fluid to assess that point fully at this time. The most that can be said

[155] Mann, supra note 1.

is that there is every reason to believe that the optimal allocation of the different models depends on the specific technology and markets involved.

Similarly, although the open source model seems to have much to lose from the patent system, it is far from clear that it would work without it. Many of the principal participants are large patentees. Those firms continue to develop proprietary hardware and software products. Patents are an important way to protect the underlying R&D investments, and increasingly are used to generate licensing revenues. The open source movement, in turn, depends heavily on the involvement of commercial participants for legitimacy in the eyes of enterprise users.

In the end, it seems certain that the different models will be forced to co-exist, in a world in which property rights will continue to matter. In addition, if they continue to co-exist, the industry will develop in a different shape than it would without the two models. I argue here that the industry will be more concentrated and harder to enter. I may be wrong about that. But if I am right, the rise of commercial open source will have an effect far different from the vision of its creators.