2005

The Commercialization of Open Source Software: Do Property Rights Still Matter?

Ronald J. Mann
Columbia Law School, rmann@law.columbia.edu

THE UNIVERSITY OF TEXAS SCHOOL OF LAW

Harvard Journal of Law & Technology, Vol. 20(1) (2006)

Law and Economics Research Paper No. 58
September 2006

The Commercialization of Open Source Software: Do Property Rights Still Matter?

Ronald J. Mann

The University of Texas School of Law

Harvard Journal of Law & Technology
Volume 20, Number 1 Fall 2006

COMMERCIALIZING OPEN SOURCE SOFTWARE: DO PROPERTY RIGHTS STILL MATTER?

*Ronald J. Mann**

TABLE OF CONTENTS

I. INTRODUCTION

For several years now, open source software products have been gaining prominence and market share. Yet the products themselves are not as provocative as the way in which they are developed and

* 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 Peggy Radin, Pam Samuelson, and the participants in the Intellectual Property workshop at Boalt Hall, for useful comments on an earlier draft.

distributed. Two related features of the open source model are distinctive: the use of collaborative development structures that extend beyond the boundaries of a single firm, and the lack of reliance on intellectual property ("IP") rights as a means of appropriating the value of the underlying technologies. Firm-level control of intellectual property is replaced by a complex set of relations, both informal and sometimes contractual, among strategic partners not joined by 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 while 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 research and development ("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 evidence of a robust startup market and of investors' lack of concern about patents held by competitors. More generally, I argue that many of the criticisms of software patents fail to account for the potential 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 Article takes up that issue.

---

1. *See* 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*, RES. POL'Y (forthcoming), *available at* http://www.utexas.edu/law/faculty/rmann/info/Data/57.pdf (last visited Oct. 16, 2006).

In essence, 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 little regard for the possibility of patent infringement. Those two practices cannot coexist for long. If the existing legal framework is not abandoned, then the major open source projects must acquire patents of their own or they must rely on the patent portfolios held by those who participate in the proprietary model.

Yet some would say that the potential for high-quality software development through the open source model justifies eradication of software patents for the entire software industry. More formally, one potential cost of permitting ready enforcement of software patents is the disabling of the open source model. At the same time, a sensible policy analysis must consider the entrepreneurs and small firms struggling to find a foothold in the industry. The property rights that patents offer are closely connected with the survival and success of those firms. As a result, 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 them 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 — 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, 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 it is more difficult for startups entering the industry to compete in the services sector than in the products sector, this suggests in turn that the spread of open source software could promote concentration in the software industry.

This Article analyzes 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 is a brief description of the landscape of the industry

2. *See, e.g.*, Microsoft Shared Source Initiative Home Page, http://www.microsoft.com/resources/sharedsource/default.mspx (last visited Sept. 30, 2006).

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 relationship between open source and the direction and location of innovation in the industry. This Article argues that open source development is more likely to support innovation by larger and better-established firms, where proprietary development is at least *relatively* more accessible to startup and younger firms.

A Note on Sources

My account of the software 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 a broad array of news accounts relating to the open source community. Third, I have studied the texts of the actual licenses with considerable care to understand how the licenses in the industry work. Although some scholars have noted the important distinctions in these licenses,[4] the literature generally has failed to consider a link between the terms of such 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 with investors and entrepreneurs in smaller startup firms. 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, 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. Due to the nature of these concerns, and because I thought it important to obtain access to frank opinions from executives at large companies, I adopted a restrictive framework for

3. For what I consider the most noteworthy, see 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*, 19 J. ECON. PERSPS. 99 (2005); Josh Lerner & Jean Tirole, *The Scope of Open Source Licensing*, 21 J. L. ECON. & ORG. 20 (2005) [hereinafter *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. *See, e.g.,* ROSEN, *supra* note 3; WEBER, *supra* note 3; Ieuan G. Mahony & Edward J. Naughton, *Open Source Software Monetized: Out of the Bazaar and into Big Business*, COMPUTER & INTERNET LAW., Oct. 2004, at 1.

the interviews. Specifically, the interviews for this project were conducted on the understanding that (1) I would not identify the specific individuals to whom I spoke; (2) I would emphasize that the interviewees expressed their personal views rather than the views of the firms by which they were employed; (3) my notes of the conversations would remain confidential; and (4) I would not attribute any specific quotations to employees of a particular firm.[5] Because several of the firms were generous enough to provide 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 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 selected the interview base and incorporated my notes into this Article. In the end, however difficult it might be to replicate this information, it is fair to say that my approach does fall squarely within the relevant methodological tradition in the social sciences. Thus, the concerns with replicability should go to the weight to be ascribed to the information, rather than its usefulness or validity.[6]

II. THE LANDSCAPE

A summary of the development and current state of the software industry provides a necessary backdrop to the analytical questions on which this Article focuses. 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 came into existence in the mid-1960s when labor shortages made it difficult for increasingly complex software to

5. In order to ensure a better understanding of this Article, I will cite to information gleaned from these interviews as "Interviews with Software Executives" where appropriate.

6. 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 (3d ed. 2002).

be produced in-house by each computer user as needed.[7] 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.[8]

The industry has two sectors: products and services.[9] The product sector further divides into two markets, one for sales to individuals and the other to businesses (called enterprise software).[10] The enterprise software market, in turn, includes products aimed at software designers and developers,[11] products targeted directly to end users,[12] and products targeted to hardware developers.[13] The service sector is less structured. It includes everything from outsourcing the entire IT function, to maintenance contracts, to custom software design, to hosted applications delivered via a web browser. The critical distinction between the last category and a prepackaged software product may be the difference 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 the sectors 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. Services firms have lower operating margins and lower growth rates, but can more readily establish stable market positions.[14] 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. A large group of hybrid firms fall between. These firms

---

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

8. *See id.*; MICHAEL A. CUSUMANO, THE BUSINESS OF SOFTWARE (2004); VERNON W. RUTTAN, TECHNOLOGY, GROWTH AND DEVELOPMENT: AN INDUCED INNOVATION PERSPECTIVE 338–340 (2001).

9. Although the general distinction between products and services firms draws on CUSUMANO, *supra* note 8, the further breakdown in this paragraph is my own.

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

11. Examples would include web development tools, graphics tools, server software, operating systems, and firmware.

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

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

14. *See* John Allison, Abe Dunn & Ronald J. Mann, *Software Patents, Incumbents, and Entry*, 85 TEX. L. REV. (forthcoming 2007). Products firms are more likely to use patents than services firms.

generally started by attempting to sell products but subsequently were forced by market conditions to provide ever-increasing levels of customization thus degrading their ability to sell high volumes of a high-margin product.[15]

A remarkable lack of concentration characterizes the software industry as a whole. The industry's CR4 ratio[16] is only thirty-nine percent, and its HHI[17] is less than six hundred (where an HHI of one thousand or more qualifies an industry as only moderately concentrated).[18] Census Bureau statistics report more than fifty thousand firms in the industry as of 2002.[19] Last year, nearly five hundred firms in the industry had more than $1 million in sales,[20] and venture capitalists invested more than $4.7 billion in software firms.[21] 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.[22] Because the venture capital model that supports most new firms entering the industry better suits products firms than services firms,[23] products firms are more likely to receive financing. New services firms, although not unheard of, are less common and tend to evolve naturally

---

15. *See* CUSUMANO, *supra* note 8.

16. The industry CR4 is the concentration ratio of, or percentage of total market sales accounted for by, the top four firms in the industry (here software firms).

17. The Hirschman-Herfindahl Index ("HHI") assesses concentration by summing the squares of the individual market shares of all participants.

18. I calculated these measures using the 2002 software sales for firms in the Software 500, the five hundred largest software firms by revenue. My calculations 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.

19. In the most recent economic census, there were more than 9900 firms in NAICS 5112 (Software Publishers), Industry Statistics Sampler: NAICS 5112, http://www.census.gov/econ/census02/data/industry/E5112.HTM, and more than 48,000 firms in NAICS 541511 (Custom Computer Programming Services), Industry Statistics Sampler: NAICS 541511, http://www.census.gov/econ/census02/data/industry/E541511.HTM.

20. *See supra* note 18.

21. 2005 PRICEWATERHOUSECOOPERS MONEYTREE REPORT 2 (2005), *available at* http://pwcmoneytree.com/exhibits/05Q4MoneyTreeReport_FINAL.pdf.

22. In 2005, 246 software firms received their first round of venture capital financing. Between 1995 and 2005, there were nearly 3,800 first-sequence investments in the software industry. PricewaterhouseCoopers MoneyTree Report Historical Trend Data, http://www.pwcmoneytree.com/exhibits/NationalAggregateData95Q1-06Q2-FINAL.xls. Many of the firms that received financing during that period have failed. *See* Mann & Sager, *supra* note 1.

23. *See* Mann & Sager, *supra* note 1; *see also* CUSUMANO, *supra* note 8.

from incumbent or rising products firms adapting to market pressures.[24]

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 service firms that sold custom-designed products provided the source code to the user, but with restrictions designed to prevent further disclosure.[25] Until about 1990, the standard license agreements for prepackaged products generally did not make the source code available at all. This led to increasing compatibility problems between software and hardware components because software developers did not have access to one another's source code.[26]

The rise of the Internet and network computing, both of which have increased the technical complexity of software, exacerbated the interoperability problem.[27] The problem is particularly acute for infrastructure and enterprise products, as opposed to end-user applications that tend to be easier to install. The commoditization of "middleware"[28] made custom software less dominant in the enterprise space and increased the importance of easy compatibility. It is difficult to sell a commodity that does not easily interact with other commoditized products that provide associated functionality. The technical complexity of software underscored the need for transparency in software design, as many 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.[29] Although there obviously was competition among firms to own the "platform,"[30] 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.

---

24. Interviews with Software Executives, *supra* note 5.

25. *Id.*

26. *Id.*

27. *Id.*

28. By middleware, I refer to software that operates as an intermediary between different applications and provides core functionality, such as web servers, applications servers, and database management systems.

29. A platform product is a piece of software that provides functions on which other applications are based.

30. 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).

Theoretically, a workable method for top-down articulation of platform standards or interfaces could have avoided this problem. This has not happened. 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 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 knowingly has been adopted into standards in some cases, and there have been several notable incidents where patents were discovered after a standard was implemented.[31]

Others advocate the mandatory licensing of intellectual property rights incorporated into standards. They recognize the difficulty of establishing a property-free zone in which to articulate standards. 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"), generally has taken the view that participants in a standards process must contribute their patents royalty-free.[32] 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. Many patentees in the industry instead insist that a better approach is to permit a standard to incorporate patents licensed on a RAND basis even if they are not fully royalty-free.[33]

Recently, cost pressures have given open source products an important entry point into the commercial market. Among other things, those products are attractive because of their ability to facilitate lower hardware costs by preventing vendor lock-in.[34] Furthermore, sophisticated enterprises are more willing to take a risk on a potentially complex installation and integration process.[35] The early dissemination and widespread adoption of Linux and Apache — both free of cost

---

31. *See* Michael G. Cowie & Joseph P. Lavelle, *Patents Covering Industry Standards: The Risks to Enforceability Due to Conduct Before Standard-Setting Organizations*, 30 AM. INTELL. PROP. L. ASS'N. Q.J. 95 (2002).

32. *See* ROSEN, *supra* note 3, at 303–11. The Organization for the Advancement of Structured Information Standards ("OASIS") recently revised its patent policy to accommodate but not require royalty-free licenses. OASIS - Who We Are - Intellectual Property Rights, http://www.oasis-open.org/who/intellectualproperty.php (last visited Oct. 16, 2006). Even the W3C policy permits royalties through an opt-out provision. *See* W3C Patent Policy, § 7, Feb. 5, 2004, http://www.w3.org/Consortium/Patent-Policy-20040205.

33. Interviews with Software Executives, *supra* note 5.

34. *See* Eric S. Raymond, THE MAGIC CAULDRON (2000), http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/index.html.

35. Interviews with Software Executives, *supra* note 5.

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.[36]

3. Cross-Licensing: the Proprietary Equilibrium

As I have explained elsewhere,[37] 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 have made it important for all of the major firms to have access to the intellectual property of the other major firms in the industry. 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 portfolios through litigation, but instead have chosen for the most part to enter 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 the other large firms. 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.[38]

B. The Open Source Development Model

As a method of software production, open source dates back to the earliest days of commercial computing, when businesses using IBM computers in the early 1950s collaborated on the task of designing software for their machines.[39] The modern history of open source, however, begins with the birth of UNIX in 1969. Starting with a few months of programming by Ken Thompson at his California home, UNIX developed into a widely used and respected operating system

36. *Id.*

37. *See* Mann, *supra* note 1.

38. The strength of the IP portfolios remains relevant. IBM, for instance, has the strongest patent portfolio in the industry and earns hundreds of millions of dollars in licensing fees each year. As a result, other firms have an incentive to increase the strength of their portfolios to lower the net sums they must expend on cross-licensing agreements with IBM. Most other cross-licensing agreements do not involve monetary payments. Interviews with Software Executives, *supra* note 5.

39. CAMPBELL-KELLY, *supra* note 7.

that has become the ultimate source of many of the most successful operating systems in use today.[40]

For purposes of this Article, 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 offer the benefits of the UNIX operating system but include sufficient 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 more commonly in recent years, 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.[41]

As the history of Linux suggests, open source holds its greatest promise for platform products. The market need is greatest for platform products because of the importance of a reliable promise that vendor lock-in will not endanger the survival of products built or modified on the software stack above that platform.[42] It is more important in ensuring interoperability to have access to the source code of platform products on which middleware and applications must be stacked. Further, collaborative development has its highest potential in the area of platform products, 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[43] that is not, at least in its current manifestations, primarily a platform product.[44]

---

40. WEBER, *supra* note 3, at 20–53.

41. *See id.* at 54–55, 94–109.

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

43. 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 July 2006, it is estimated that Firefox has a 13 percent share of the browser market, compared to 83 percent for IE and 2 percent for Safari. Alexandra DeFelice, *Firefox Claims Bigger Chunk of Browser Market*, LINUXINSIDER, July 12, 2006, http://www.linuxinsider.com/story/51736.html.

44. To be sure, the rise of web application "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, http://news.com.com/

1. The Current State of Open Source: Commercialization

Some open source programs are created almost entirely through the efforts of volunteers,[45] as in the early days of Linux. As others have recognized, a key part of an open source project is attracting talented and motivated individuals.[46] Even now, most important projects have roots in self-organized collaborative activity.[47] Yet the events of the last few years show that the ties between open source communities and large incumbent (proprietary) firms are increasing rapidly. A substantial share of the important Linux contributors now have gainful employment either directly for the Open Source Development Labs ("OSDL") or for one of its major supporters.[48] 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.[49] Moreover, dual-licensing firms[50] (like MySQL) generally directly employ almost all of those who contribute to their projects. Firms can reject any code submitted by individuals who 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.

Mapping+a+revolution+with+mashups/2009-1025_3-5944608.html; Ryan Singel, *Are You Ready for Web 2.0?*, WIRED NEWS, Oct. 6, 2005, http://www.wired.com/news/technology/0,1282,69114,00.html.

45. 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 Scope of Licensing*, *supra* note 3 (analyzing SourceForge data).

46. *See* ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR (1999); Yochai Benkler, *Coase's Penguin, or, Linux and the Nature of the Firm*, 112 YALE L.J. 369 (2002); Jeffrey A. Roberts et al., *Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects*, 52 MGMT. SCI. 984 (2006); *see also* Justin Pappas Johnson, *Open Source Software: Private Provision of a Public Good*, 11 J. ECON. & MGMT. STRATEGY 637 (2002); Dan M. Kahan, *The Logic of Reciprocity: Trust, Collective Action, and Law*, 102 MICH. L. REV. 71 (2003); Raymond, *Magic Cauldron*, *supra* note 34.

47. In recent years, proprietary companies have tried, with varying levels of success, to start projects by releasing proprietary code as open source software. Examples include Netscape's release of its browser source code in 1998 which eventually formed the basis of Firefox, IBM's 2004 release of Cloudscape to the Apache Foundation, and Sun's release of the source code for Solaris 10.

48. *See* Daniel Lyons, *Peace, Love, and Paychecks*, FORBES, Sept. 20, 2004, http://www.forbes.com/forbes/2004/0920/180.html (discussing corporate sponsorship of key Linux contributors).

49. *Cf.* Keith Regan, *Browser Rumors Renewed as Google Hires Firefox Programmer*, E-COMMERCE TIMES, Jan. 25, 2005, http://www.ecommercetimes.com/story/40015.html (reporting that Google hired the developer most responsible for the Firefox browser).

50. Dual licensing firms are firms that use separate licenses for two closely related products, one open source and one proprietary.

The availability of venture financing affects the ways open source firms enter the market. It is difficult to obtain financing for a product that will be distributed without charge, for which the source code will remain open if the product succeeds, and which (like all software products) may never succeed for technical or market-based reasons. Although some of the interviewees suggest that there are a "huge number" of startups building on Linux, it is not clear how to evaluate that suggestion. To gain perspective, I used VentureXpert[51] to identify venture capital investments in firms that develop open source software. As of August 2006, I found deals involving 135 U.S. firms — about eighty percent of which are in the software and Internet sectors — whose business descriptions contain the terms "Linux," "Apache" or "open source." By any standard, that is a small part of the relevant startup market. By comparison, nearly 8000 firms have received venture financing in the software and Internet sectors between 1998 and August 2006 (the period during which ninety-five percent of the open source investments were made). It appears that few of those firms are actually profiting directly from open source technology. Many simply offer heterogeneous (or cross-platform) operating system support, including Linux or Windows,[52] or provide proprietary applications that can be used on either a Windows or Linux platform,[53] or, on occasion, provide proprietary applications that can be used only on a Linux platform.[54]

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 running different operating systems.[55] Open Source Risk Management sells legal protection against copyright and patent infringement litigation related to open source products.[56] Finally, some 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.

---

51. VentureXpert is a source for information about venture capital and private equity funds. VentureXpert, http://www.venturexpert.com (last visited November 13, 2006).

52. *E.g.* Mission Critical Linux.

53. *E.g.* Altiris, Atempo, and PERSIST Technologies.

54. *E.g.* Aduva, Eazel, Eternal Systems, Qlusters, and Scalix.

55. *E.g.* Cassatt, Centrify, Steeleye Technology, and Vintela.

56. There are other firms that are not focusing on 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.

This 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. 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, this is a difficult route, but it may not be an impossible one.

Similarly, a few firms have obtained venture financing after their open source product was distributed, modified, and already a market success. 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 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.[57]

Another possibility is to start with open source code as the platform on which to build a proprietary product. Several venture capitalists suggest that this type of startup is increasingly common.[58] The basic expectation here is that the startups will build proprietary products on open source platforms, and that the open source nature of the platform will make it easier for the startup to integrate its work with the platform. As time goes on, it may well be that this will become an increasingly common method for the development of proprietary software. That type of development is still at an early stage. For now, an open source foundation is still likely to be an obstacle to sophisticated venture-backed financing.

2. Software Products Licenses Under Open Source Models

At the center of all of this is the license that governs the use of the code. Perhaps the most important distinction between proprietary and open source licensing involves the parties to the licenses. For the proprietary products discussed in the preceding section, the arrangement is simple: a license runs directly from each developer that publishes a

57. *See* Mann & Sager, *supra* note 1.

58. *See* Martin LaMonica, *Open Source, Open Wallet*, CNET NEWS.COM, Nov. 7, 2005, http://news.com.com/Open+source,+open+wallet/2100-7344_3-5934144.html (discussing venture capital investments in open source-related startups).

product to each end user of the product. For open source projects, however, the arrangements are more complex, with two separate stages of licensing. 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 one of several nonprofit entities (such as the Free Software Foundation) that acquires the copyright through assignment. When the copyright is not assigned, the contributor typically licenses the code to the project under the relevant license.[59] Second, in the distribution stage, the software product is distributed under the terms of an open source license. This license restricts the rights of the user in the code.

To qualify as an open source license, a license must have Open Source Initiative ("OSI") certification.[60] To become certified, a license must meet a set of minimum requirements designed to ensure that software is distributed with its source code and that it is reasonably available without constraint to developers and users who wish to use or modify it for their own purposes.[61] 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.[62] Throughout the 1990s, however, prior to the development of such shared source programs, the absence of any response to those issues 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 source rather than proprietary platforms.

---

59. This can create difficulties when those operating the project later wish to alter the license under which the product is distributed (known in the industry at "reversioning"), because they are likely to need consent from the original contributors.

60. The OSI is a non-profit organization founded in 1998 by Bruce Perens and Eric Raymond. Generally, it supports a broad conception of open source software that is more tolerant of commercial interaction than the Free Software Foundation's conception. For my purposes, the most important of the OSI's activities is its promulgation of the Open Source Definition, the generally accepted indicator that a particular license should be regarded as "open source." *See* OSI — The Open Source Definition, http://www.opensource.org/docs/definition_plain.php (last visited October 11, 2006).

61. Version 1.9 of the Open Source Definition includes the following requirements: free redistribution must be tolerated; source code must be included; 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. *Id.*

62. Indeed, Microsoft's shared source program is designed to address these ends. *See* Microsoft and Open Source (Oct.18, 2005), http://www.microsoft.com/resources/sharedsource/Articles/MicrosoftandOpenSource.mspx.

Aware of those issues, open source communities have established a baseline, 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 affect the commercial development of the licensed software.[63] For the present discussion, the licenses differ most importantly in three ways: (1) the constraints on incorporation of the licensed code in later products; (2) the rules about the contribution of IP rights related to contributed code; and (3) the rules about enforcement of IP rights by users of the software.[64]

The first has traditionally been the major point of differentiation among open source licenses. There is a readily discernible continuum, from "reciprocal" licenses (like the GPL) at one end to "academic" licenses (like 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."[65] Thus, the license reflects a concept of reciprocal obligation. Developers are free to take advantage of the contributions reflected in an existing piece of GPL code, provided they make a reciprocal contribution of their modifications under the GPL model.[66] The scope of restric-

63. 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, GNU Lesser General Public License ("LGPL"), Berkeley Software Distribution ("BSD"), and MIT licenses dominated significant projects until the late 1990s, but starting with the release of Mozilla in 1998 the number of licenses approved by OSI has increased rapidly. As I write, fifty-eight 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 as it is that the open source community as a whole will become a number of effectively separate gated communities. *See id.* at 247–53.

64. This section draws heavily on the terminology and analysis of ROSEN, *supra* note 3.

65. The GNU Public License § 2(b) (June 1991), http://www.opensource.org/licenses/gpl-license.php. A similar provision appears in § 5 of the Second Discussion Draft of GPLv3, which is currently under consideration by the Free Software Foundation. GPLv3, 2nd Discussion Draft (July 27, 2006), http://gplv3.fsf.org/gpl-draft-2006-07-27.html.

66. I do not address here whether the licenses are binding as a matter of contract or through rules of property rights. On that point, Peggy Radin has suggested in conversations with the author 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 where stopping subsequent use is not an adequate remedy. The main example of

tions imposed by that provision is debatable,[67] but it certainly imposes at least some constraint on the ability of a developer to incorporate GPL code into a fully proprietary product.[68] If anything, it appears that the pending revisions to the GPL will exacerbate the problem rather than mitigate it. Among other things, it seems likely that the next version of the GPL will forbid the integration of GPL-licensed software into products that employ digital rights management technology, largely because of a philosophical objection that Free Software Foundation principals have for such technology.[69]

At the other end of the spectrum, "academic" licenses like the BSD license impose no such 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.[70] For example, Microsoft easily can, and does, include some BSD code in its operating system.[71] 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:

> 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

this is likely to be in the provisions that purport to govern enforcement of patent rights by users of the software.

67. *Compare*, *e.g.*, James V. Delong, *The Enigma of Open Source Software* (Version 1.0), A1–A13 (Mar. 2004), http://www.pff.org/issues-pubs/pops/pop11.8opensource.pdf (a highly expansive interpretation) *with* ROSEN, *supra* note 3 (a much narrower interpretation).

68. Typical reciprocity provisions apply only when the work is "distributed." *See* The GNU Public License, *supra* note 65. 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 (Version 2.0) § 2.2 (Aug. 6, 2003), http://www.opensource.apple.com/apsl; Real Networks Public Source License Version 1.0 § 2.1(b), http://www.opensource.org/licenses/real.php (last visited Oct. 11, 2006); Open Software License (Version 2.1) § 5 (2004), http://www.opensource.org/licenses/osl-2.1.php; *see also* ROSEN, *supra* note 3, at 193–95.

69. *See* David Berlind, *Controversy over GPLv3 Draft Reflects the 'Incompatibility' of DRM with Open Source*, ZDNET, July 28, 2006, http://blogs.zdnet.com/BTL/?p=3399.

70. The concept behind that license resonates strongly with the academic community's notions of motivation and intellectual contribution. *See* Rebecca S. Eisenberg, *Academic Freedom and Academic Values in Sponsored Research*, 66 TEX. L. REV. 1363 (1988); Rebecca S. Eisenberg, *Proprietary Rights and the Norms of Science in Biotechnology*, 97 YALE L.J. 177 (1987).

71. Interviews with Software Executives, *supra* note 5.

> the requirements of this License are fulfilled for the Covered Code.[72]

Sun's new Common Development and Distribution License ("CDDL"), which governs its contribution of Solaris, includes a substantially identical provision (§ 3.6).[73] 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]."[74]

The second crucial point of differentiation among the licenses is the coverage of intellectual property rights held by those who contribute to the project. The traditional practice has been to rely on the understanding that any party who contributed to an open source project would grant an implied license that permitted ordinary uses of the resulting software. Licenses like the GPL[75] and the BSD that do not explicitly deal with the intellectual property rights retained by contributors must rely on that concept.[76] Recent licenses deal with the subject more directly, requiring specific copyright and patent licenses from all contributors to all users.[77] Indeed, the Apache Software Foundation has developed a separate Apache Contributor License Agreement designed specifically to respond to this problem.[78] The proposed GPLv3 takes a slightly different tack, including a covenant not to assert patent claims as opposed to a traditional license.[79]

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

---

72. Mozilla Public License § 3.7, http://www.mozilla.org/MPL/MPL-1.1.html (last visited Oct. 16, 2006).

73. Common Development and Distribution License Version 1.0 § 3.6 (Dec. 17, 2004), http://www.sun.com/cddl/cddl.html.

74. Apache License, Version 2.0 §§ 2–3 (Jan. 2004), http://www.apache.org/licenses/LICENSE-2.0.

75. GPL, *supra* note 65. Section 7 does include a 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.

76. As Rosen explains, there are numerous technical problems with relying on implied licenses, such as 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.

77. *See, e.g.*, Apache License, *supra* note 74, §§ 2–3; Common Development and Distribution License, *supra* note 73, §§ 2.1–2.2.

78. *See* Apache Software Foundation Individual Contributor License Agreement V2.0, http://www.apache.org/licenses/icla.txt (last visited Oct. 16, 2006).

79. *See* GPLv3, 2nd Discussion Draft, *supra* note 65, § 11.

patents that are necessary to the use of the Original Code.[80] 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 later, nothing in the MPL would require Netscape to license that patent to subsequent users of the code.[81] A slightly different twist comes from IBM's Common Public 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.[82]

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 is often a distributed body of difficult-to-identify contributors.[83] Open source licenses generally impose the infringement risk on licensees.[84] 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.[85] 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.

What is most interesting is the great variation in the provisions focusing on third party IP, primarily because it suggests more 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

---

80. 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. *Cf.* ROSEN, *supra* note 3, at 147–50.

81. *See id.* at 148–150. This is similar to the definition of "essential patent claims" that is used in § 11 of the proposed GPLv3. *See* GPLv3, 2nd Discussion Draft, *supra* note 65, § 11.

82. *See* Common Public License § 2, http://www.opensource.org/licenses/cpl1.0.php (last visited October 11, 2006); ROSEN, *supra* note 3, at 163–66.

83. This 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.

84. 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. *See* Mozilla Public License, *supra* note 72, § 3.4(c); *see* ROSEN, *supra* note 3, at 158, 198–201.

85. *See, e.g.*, Apache License, *supra* note 74, § 3; MPL, *supra* note 72, § 8.2. As a related matter, licenses also often require contributors to include notice of patent problems of which they might be aware. *See* MPL, *supra* note 72, § 3.4.

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.[86] 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."[87]

Those provisions have a fascinating effect because they generally operate not only to protect the products in question, but also to slowly bring open source products within the cross-licensing equilibrium that has provided stability to the proprietary segment 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 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.[88] 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.[89]

The most interesting recent development is the proposed extension in GPLv3 of the patent retaliation provision. It would prohibit a company not only from distributing GPL software, but also from using modified versions on their own servers. For example, assume that a company like HP prepares a modified version of a GPL program, which it runs on its own servers but does not distribute, and that HP obtains patents that protect the functionality that HP has added. Under the current GPL, HP would be free to assert those patents against any later innovator that attempted to incorporate the patented functionalities into the GPL program. However, under GPLv3, any such suit would vitiate HP's right to run its modified version of the GPL program on its own servers.[90] It should be no surprise that this provision

86. *See* ROSEN, *supra* note 3, at 155.

87. Apple Public Source License, *supra* note 68, § 12.1(c).

88. Open Software Library, *supra* note 68, § 10; *see also* Apache License, *supra* note 74, § 3 (containing a similar provision).

89. *See* ROSEN, *supra* note 3, at 217–18.

90. *See* GPLv3, 2nd Discussion Draft, *supra* note 65, § 2. The rationale described in the text is set out in a separate document. *See* Opinion on Patent Retaliation, http://gplv3.fsf.org/patent-dd2.html (last visited Oct. 11, 2006).

is controversial, drawing stern criticism not only from HP, but also from as venerable a figure as Linus Torvalds.[91]

III. MOTIVATIONS FOR THE COMMERCIALIZATION OF OPEN SOURCE

A. Open Source as a Viable Business Model

Open source development is aptly viewed as a direct challenge to the traditional "one shop" model of proprietary software development. The question is 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, funding, and direction of research and development are accomplished within the boundaries of a single firm. The advantages of the model are the same as those of bringing any complex activity within the boundaries of a single firm: the firm is able to collect resources from investors and then decide 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 is a strong advantage of the proprietary model.[92]

In the open source model, by contrast, control is more diffuse, with development proceeding through relatively decentralized hierarchies.[93] The strength of open source development is its potential to produce products with a higher quality and more innovative character than similar proprietary products. Discourse from supporters often reflects a deep-seated, at times almost mystical, conviction that collaborative development is superior to centrally directed development. The argument resonates strongly with the recent literature on open innovation.[94] 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 populist reveling in the idea that unsupported individuals can produce software of a commercial quality

---

91. *See* Geoffrey Lewis, *Linus Torvalds Not Happy with Revised GPL*, EARTHTIMES.ORG, Jul. 31, 2006, http://www.earthtimes.org/articles/show/7882.html.

92. Interviews with Software Executives, *supra* note 5.

93. As Weber explains in detail, there is a great deal of organization of open source development. *See* Weber, *supra* note 3. 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.

94. *See, e.g.,* HENRY WILLIAM CHESBROUGH, OPEN INNOVATION: THE NEW IMPERATIVE FOR CREATING AND PROFITING FROM TECHNOLOGY (2003).

that can compete with the output of the world's largest corporations.[95] To be sure, it is difficult to obtain empirical evidence about quality, and the existing evidence seems ambiguous.[96] The widespread adoption of commercially successful open source products offers strong testimony that in some contexts 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 will be unable to obtain 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 prohibiting 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 importance of property rights to the proprietary software industry can be overstated. Many firms do not exploit their patents, and relatively few exploit their patents to collect licensing revenues.[97] One industry executive illustrated that point effectively when he explained that in large patent-sophisticated firms in the software industry there is a ratio 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, is about 75:25.[98] To the extent that firms do collect licensing revenues,[99] 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 is more robust in a model with property rights than it is in an open source model in which the standard OSI requirements make it impractical to exclude third parties from exploitation of technology created in an open source community.

---

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

96. *See* Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265 (2004). Open source proponents can point, among other things, to the low cost of their products (which are 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. My impression is that the studies as a group are ambiguous, suggesting that one type of software might be cheaper in one context, but that broad general claims of superiority are difficult to sustain.

97. *See* Mann, *supra* note 1.

98. Interviews with Software Executives, *supra* note 5.

99. IBM collects billions of dollars each year from the licensing of software-related patents. Mann, *supra* note 1, at 997. Other incumbent firms 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, at 1006–07), it remains to be seen whether this program will generate significant revenues.

1. Predatory Motive: the "Kill Microsoft" Approach

In discussions about the economics of open source, one of the most prominent ideas is that the model itself cannot be made profitable, but that firms invest in it solely because it decreases the monopoly power of Microsoft.[100] In its simplest form, the idea is that firms are willing to invest in software development that will not generate a monetary return because of the likelihood that their efforts will lessen the ability of Microsoft to extract future monopoly profits in markets in which those firms might participate. A large customer like Intel might want 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 a new market in which Microsoft is less 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.[101] However, several software executives to whom I have spoken have emphasized that the most obvious victims of IBM's Linux strategy, to the extent that there have been victims, are UNIX competitors like Sun Microsystems, not Microsoft.[102] Sun directly competed 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).[103] Thus, predatory-motive theory seems at best an incomplete story.

2. Traditional Profit Motive: the Value Chain Approach

Although there is surely 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 are. Executives have

100. Merges, *supra* note 3, at 192–93.

101. Interviews with Software Executives, *supra* note 5.

102. *Id.*

103. The competition between IBM and Sun is to some degree bound up in their differing open source strategies. IBM 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* Raymond, *supra* note 34 (arguing that Sun's license structures have alienated open-source communities).

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 group of seven large proprietary companies that are major investors in the OSDL: IBM, HP, Intel, Fujitsu, Red Hat, Novell, and General Motors.[104]

The most logical explanation for those investments comes from the value chain concept. 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 a 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 higher returns for those portions of the stack in which it can compete.

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 Dell's increasing market domination illustrates the success of its focus on its core competency in logistics.

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, this is no different from a developer dedicating public streets in a subdivision to maximize the total value of the development.[105] Just as all homeowners in an area can benefit

---

104. 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 permit their employees to write open source code 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 and Ziedonis show, the investment models for those investments are quite difficult to understand. *See* David Benson & Rosemarie Ziedonis, *Don't Fence Me In: Fragmented Markets for Technology and the Patent Acquisition Strategies of Firms*, 50 MGMT. SCI. 804 (2004). In this context, I expect that it is difficult to quantify the likelihood that a firm would support an open source startup that itself might never be profitable, but would increase demand for hardware, services, or infrastructure products sold by the sponsor. This surely explains why Intel Capital is the most prolific investor in the open source-related startups I discuss above. It invested in fourteen of the sixty-six United States firms in the "Computer Software" sector. *See* VentureeXpert.com Home Page, http://venturexpert.com.

105. *See* Oren Bar-Gill & Gideon Parchomovsky, *The Value of Giving Away Secrets*, 89 VA. L. REV. 1857, 1973–74 (2003) (exploring why an innovator might gain more profit

from sharing a single public street that runs near all of their homes, OSDL members benefit by sharing the costs of production of the Linux operating system. One group of executives suggested that maintaining a competitive enterprise software platform currently requires about $500 million of investment each year.[106] If IBM can spend $100 million per year on Linux and obtain access to such a platform, that is much cheaper than maintaining the platform on its own.[107]

Thus, investing in Linux is a rational step for the individual members of the OSDL not because it might harm Microsoft or generate profits from direct sales, but because developing Linux as a high-quality operating system permits each of them to develop complementary goods and services in their respective core competencies.[108] From this perspective, Linux is similar to other products that can be used as platforms for third-party software. Other vendors may find this value chain preferable to the competing Microsoft/Intel value chain because in the former, the operating system cannot be used to extract profits. Indeed, open source software is optimally suited for this type of arrangement. It is the ultimate commodity: anybody can distribute it at no cost to the end user, promoting cooperation between market players and making it difficult for a single firm to develop a dominant position to undermine the OSDL strategy.

Moreover, when IBM and other members of the OSDL began investing in Linux, the operating system was already making inroads in the server market.[109] If those companies had resolutely stayed outside that field, they would have 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 still may happen to Sun despite its efforts to participate in the open source community. The optimal response to that situation is to attempt to co-opt the potentially

---

from an innovation if it could foster related innovations through a gift to the public domain of some portion of the innovation).

106. Interviews with Software Executives, *supra* note 5.

107. *See* Raymond, *supra* note 34 (discussing the benefits of cost-spreading).

108. Mahony & Naughton, *supra* note 4; Bruce Perens, *The Emerging Economic Paradigm of Open Source*, FIRST MONDAY, Oct. 3, 2005, http://www.firstmonday.org/issues/special10_10/perens/index.html. Economists that have examined the question have concluded that in some circumstances the profits from applications built on an open source platform can be greater than the profits from a set of proprietary applications and platform. Nicholas Economides & Evangelos Katsamakas, *Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry*, 52 MGMT. SCI. 1057, 1058 (2006).

109. *See generally* P.G. Capek et al., *A History of IBM's Open-Source Involvement and Strategy*, 44 IBM SYSTEMS J. 249 (2005) (presenting an official IBM account of its involvement and strategy).

disruptive technology into the business model of the existing firm.[110] However, this does not stop the disruption. Rather, as suggested above, it focuses the disruption on the firms least capable of integrating the new technology into their business models (namely Sun, if this analysis turns out to be correct).[111] Thus, investment in open source has been successful as a disruptive strategy.

To use an obvious example, IBM is one of the most multi-faceted firms in the IT industry. Even if IBM cannot profit from sales of the Linux operating system and the Apache web server program, it can profit by offering a value chain that uses those programs. First, 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 retains major hardware lines in the areas where Linux is most commonly used. Second, IBM can write proprietary software that can be used on those computers. For example, after IBM failed to write its own successful web server, it surrendered to the dominant Apache program. It then developed its highly successful WebSphere program, which is designed specifically to run on computers that use Apache. Offering software designed for the large community of firms already using Apache was key to gaining market adoption and a marked improvement on IBM's earlier efforts to bundle similar products with its own proprietary server programs. Third, IBM is an industry leader at providing services that integrate various hardware and software products which is one of its most profitable enterprises.[112]

Apple's deployment of Mac OS X is another 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 high quality

---

110. *See generally* CLAYTON M. CHRISTENSEN, 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 as a high-quality flexible product, moving from the most demanding users to the least demanding, rather than moving from the least demanding users to the most demanding. Interviews with Software Executives, *supra* note 5.

111. 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. Interviews with Software Executives, *supra* note 5.

112. My analysis is not undermined by the examples in Peter Swire's cogent article on the security market. *See* Peter P. Swire, *A Theory of Disclosure for Security and Competitive Reasons: Open Source, Proprietary Software, and Government Systems*, 42 HOUS. L. REV. 1333, 1356 (2006) (discussing an earlier draft of this Article). 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 that an undue level of indirectness and complication is necessary for proprietary firms to profit in this area. *Id.* 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 — are precisely the type of business models that I discuss here.

graphical user interface, thereby focusing proprietary efforts on one of Apple's core competencies.

B. Open Source as a Market Correction

The most thoughtful assessment of the role of IP in the open source context is Rob Merges's *A New Dynamism*,[113] which generally portrays open source as a market correction responding to excessive protection of IP. Merges views the investments that proprietary firms make in open source projects as "property preempting investments" ("PPI") — or a form of "anti-property." Those investments are designed to protect the commons from enclosure by IP rights held by incumbents (of whom Microsoft is Merges's principal concern). Although that perspective brings a healthy dose of economic analysis to a subject that is often unduly romanticized,[114] 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 creating "anti-property" rights.[115] To paraphrase his argument, the investments are designed to make an "exclosure" — the opposite of an enclosure — as a "property-free zone"[116] into which later actors cannot force their proprietary claims. That is not, however, a complete answer. To be sure, developers write and contribute code to a community under broad licenses. For several reasons, however, that does not have nearly so bucolic an effect as the casual reader of Merges's paper might assume.

The first reason is the simplest one: contributors to open source projects for the most part do not convey their IP rights wholesale to the open source community.[117] Instead, contributors may retain own-

---

113. Merges, *supra* note 3.

114. *See also* Anupam Chander & Madhavi Sunder, *The Romance of the Public Domain*, 92 CAL. L. REV. 1331 (2004).

115. Merges, *supra* note 3.

116. Further, the rhetoric of a commons 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 NEWS.COM, Aug. 22, 2005, http://news.com.com/Torvalds+weighs+in+on+Linux+trademark+row/2100-7344_3-5841222.html (discussing Linus Torvalds's defense of the vigorous action taken on his behalf to enforce the Linux tradename); Ingrid Marson, *JBoss Denies Running a Trademark Monopoly*, CNET NEWS.COM, Oct. 11, 2005, http://news.com.com/JBoss+denies+running+a+trademark+monopoly/2100-7344_3-5893015.html (Marc Fleury's response 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 this Article, it is still hard to say that property rights are not important in open source.

117. This is by no means universal. Many contributors do, in fact, convey their rights to entities like the Free Software Foundation or the Apache Foundation, which for my purposes would seem to be trustees of the "exclosed" commons.

ership of the IP rights in the code they create and merely license those rights to the open source community. In the case of Linux, hundreds of contributors own copyright interests in their contributed code and thereby can prevent Linux from adopting the new version of the GPL. Since Linux is the largest open source project released under the GPL, those contributors in fact retain some ability to hinder reversioning of the GPL.[118] The possibility of conflict is real: 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.

A disagreement over the direction a project should take will ultimately be resolved in favor of the person who controls the relevant IP (whether it be copyrights in the source code, control of the trade name, or ownership of important patents).[119] Similarly, reversioning of the GPL would be easier if every Linux contributor would agree to anything the Free Software Foundation and OSDL submit as an appropriate update of the GPL.[120] 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's control of much of the core IP in Linux that gives him so much negotiating power in the struggle to update the GPL.

Additionally, consider the case of dual-licensing firms like MySQL, where the firm that employs a project's contributors holds substantially all of the IP rights to that project. This dual-licensing structure allows the firm to use a conventional proprietary licensing model to profit from a version of the software that might not differ 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 (MPL, Apple, and Sun licenses, etc.), 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.[121] 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.

---

118. For discussion of the reversioning effort, see Welcome to GPLv3, http://gplv3.fsf.org (last visited Oct. 4, 2006) .

119. *See, e.g.*, Stephen Shankland, *Open-Source Mambo Project Faces Rift*, CNET NEWS.COM, Aug. 22, 2005, http://news.com.com/Open-source+Mambo+project+faces+rift/2100-7344_3-5841347.html (discussing dispute among contributors to Mambo).

120. The possibility of conflict is becoming more serious, as Linus Torvalds has announced his dissatisfaction with the early drafts of GPLv3.0. *See* Lewis, *supra* note 91.

121. 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, at 126.

Rather, they are limited to existing patents, 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 when contributors have used licenses or contribution agreements that 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. Open source releases might amount to a sufficiently public use of the code to constitute prior art, thus preventing others from obtaining subsequent patent rights. They would not, however, prevent the assertion of patent rights by persons 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 that it owns all of the patents necessary for the deployment of Solaris.[122] Early and aggressive patenting can make it difficult for independent designers to obtain patents directed to their products. Even there, the possibility of bombshell patents is real in light of the high pace of innovation, where foundational patents could easily issue in 2005 for technology first invented in 2001. 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 certain patents against potential infringers, none of them has made a binding commitment to forgo their enforcement rights entirely. To the contrary, patent rights are maintained as part of the elaborate equilibrium of cross-licensing arrangements.

In 2005, several major players such as IBM, Sun, and Nokia issued pledges not to enforce their patents.[123] Those statements, however, did not contribute the patents to a commons, much less to a property-free public domain. For example, IBM's pledge was made to developers of open source products and not to the public at large.[124]

---

122. This claim seems most implausible, although it has been made quite publicly. *See* Sun Grants Global Open Source Community Access to More than 1,600 Patents, http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.2.xml (last visited November 8, 2006).

123. 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 IBM is forced to defend itself. That statement broadly covers the entire portfolio, but is unlikely to create reliably enforceable obligations on the part of IBM as circumstances change in the ever-developing landscape of the industry.

124. 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." IBM Statement of Non-Assertion of Named Patents Against OSS, http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf (last visited Oct. 16, 2006). The patents 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

Further, the underlying technology is not available for the development of proprietary offerings by competing products or services firms (such as Microsoft and 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,[125] so it does little more than the grant of patent rights that would be included in a license to use Solaris.[126] Responding to a barrage of criticism regarding 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."[127] Still, it is not clear that Sun will not enforce its patent portfolio to challenge Linux as a competitor to Solaris. To the extent that 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 even clearer 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.[128]

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: it is not constructive to think of these investments as creating a truly open domain, or in Merges's terms, a "property preempting investment."

Still, there is little doubt that open source strategies are deterring others from enforcing their patent rights in some contexts. These strategies are similar to, but potentially more powerful than, the creation of large patent portfolios within individual firms. Using combined patent portfolios to create fences around some open source

---

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. *Id.*

125. Sun's pledge purports to give free access to patents "under the Common Development and Distribution License (CDDL)." *See* Sun Grants Global Open Source Community Access to More than 1,600 Patents, *supra* note 122.

126. Common Development and Distribution License, *supra* note 73, § 2.1(b). As discussed above, provisions to that effect are ubiquitous in modern open source licenses.

127. Stephen Shankland, *Sun: Patent Use OK Beyond Solaris Project*, CNET NEWS.COM, Jan. 31, 2005, http://news.com.com/Sun+Patent+use+OK+beyond+Solaris+project/2100-7344_3-5557658.html.

128. *See, e.g.,* Jim Wagner, *Nokia's Linux Pledge*, DEVELOPER, May 26, 2005, http://www.internetnews.com/dev-news/article.php/3508146. For an additional anecdote about Computer Associates, compare the laudatory press release discussing the pledge by Computer Associates, Chris Preimesberger, *CA Patents Made Available to Open-Source Community*, EWEEK, Sept. 7, 2005, http://www.eweek.com/article2/0,1895,1856420,00.asp (a press release lauding Computer Associates' pledge) with Matt Whipp, *Computer Associates' Patent Donation Is Slammed*, PC PRO, Sept. 13, 2005, http://www.pcpro.co.uk/news/77337/computer-associates-patent-donation-is-slammed.html (criticizing Computer Associates' pledge on the grounds that the covered patents were worthless or irrelevant to the open source community).

technologies, the large firms are shifting the equilibrium to send 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."[129]

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 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 fundamentally weakening 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[130] for expecting that the prevailing open source business models will have consequences on the location of innovation.[131] As Tim Wu explains, there is good reason to think that this kind of effect — an effect on the "decision architecture" of an industry — will often be a more important effect on intellectual property rights than a direct effect on competition caused by exploitation of the right to exclude.[132]

I start with the theory articulated by Ashish Arora and his coauthors that a stronger IP system often leads to smaller and more spe-

---

129. 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*, ZDNET, Dec. 10, 2005, http://news.zdnet.com/2100-3513_22-5943781.html. If the licenses gain broad acceptance, this could lead to a shared equilibrium for the patents held by those entities. As of October 2006, however, the web site for Open Invention Network indicated that it had acquired only thirteen patents. Open Invention Network, http://www.openinventionnetwork.com/pat_owned.php (last visited Oct. 16, 2006).

130. *See* Petra Moser, *How Do Patent Laws Influence Innovation? Evidence from Nineteenth-Century World Fairs*, 95 AM. ECON. REV. 1215 (2005) (reporting empirical evidence that stronger IP systems influence the direction of innovation). The recent history of the software industry, which has seen a great deal of innovation as software patents have become easier to obtain, illustrates this contention.

131. 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.

132. Tim Wu, *Intellectual Property, Innovation, and Decision Architectures*, 92 VA. L. REV. 123 (2006).

cialized firms.[133] They reason that strong IP rights generally encourage investment in specialized firms with a superior ability to innovate, largely because strong IP rights limit the costs of leakage that occur when the locus of innovation is beyond a firm's boundaries.[134] Conversely, a weaker IP system makes it more difficult to protect proprietary technology and thus prompts the creation of larger firms and industry consolidation.[135] 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 industry and the semiconductor industry provides empirical support for that possibility.[136]

The theory that Arora and his coauthors have articulated has obvious applications to the software industry. There, innovation is cumulative because 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 the case that use of property rights to codify 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 contribute to agreements with their various cross-licensing partners, or to exclude from the equilibrium firms that do not contribute their share of innovation.

As property rights were strengthened in the mid-1990s, the software industry became increasingly fragmented. It is therefore possible that fragmentation has supported a higher rate of innovation than otherwise would have existed. The natural question, then, is whether open source will alter 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.[137] This

---

133. Ashish Arora & Marco Ceccagnoli, *Patent Protection, Complementary Assets, and Firms' Incentives for Technology Licensing*, 52 MGMT. SCI. 293 (2006); Ashish Arora & Robert P. Merges, *Specialized Supply Firms, Property Rights and Firm Boundaries*, 13 INDUS. & CORP. CHANGE 451 (2004).

134. *See* Arora & Ceccagnoli, *supra* note 133; Arora & Merges, *supra* note 133.

135. *See* Arora & Ceccagnoli, *supra* note 133; Arora & Merges, *supra* note 133.

136. *See* Ashish Arora, *Patents, Licensing, and Market Structure in Chemicals*, 26 RES. POL'Y 391 (1997); *see also* 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); Benson & Ziedonis, *supra* note 104.

137. Although it is more difficult to quantify effects on the level of innovation, the rise of open source could have effects there as well. My sense is that corporate participation in the movement 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

is true 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.

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 product in a number of important ways. For instance, one may expect 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.[138]

It may be that proprietary software developers do not often incur ironclad contractual obligations on all of those points. But despite what their contracts might say, they certainly have considerable residual legal responsibility for those problems. 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 open source authors are likely to categorically disclaim responsibility for the kinds of problems discussed above.[139] That makes some sense given the nature of the software's development, where specific contributions are made by individuals who cannot expect to use profits from the sale of the software to defray anticipated liabilities that might arise from its distribution and use. Moreover, even the proprietary companies that operate in the open source community almost uniformly disclaim any legal responsibility for problems with the software.[140]

---

period of rapid fragmentation and innovation until the number of possible ways to commercialize the technologies begins to stabilize. That period would be 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, governments and R&D divisions of large firms until some new "transformative need" is identified.

138. Unlike copyright law, which does not control "use" of a copyright work, a patent controls any use of the patented technology. *See* 35 U.S.C. § 271(e)(1) (2000).

139. *See, e.g.*, Apache License, *supra* note 74, §§ 7–8; The Regents of the University of California, The 4.4BSD Copyright ¶ 2 (1994), http://www.freebsd.org/copyright/license.html; CDDL, *supra* note 73, § 5; GPL, *supra* note 65; MPL, *supra* note 72, § 9.

140. Interviews with Software Executives, *supra* note 5.

That means that the motivation behind any response to users' problems with open source software is likely to come from a reputational incentive rather than an enforceable legal obligation. It is, of course, much more difficult for a business to assess the reliability of a reputational incentive than that of 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.

A large and publicly visible firm will be more responsive to reputational incentives than a small and emerging firm, for obvious reasons. 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. Even a relatively small, publicly-traded firm like Pervasive would have an advantage in finding customers for an open source database project over smaller startup firms purveying similar products, such as Green Plum. Detractors of open source software often argue that it is risky for a business to rely on reputational considerations 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 open source software is closely associated with a large and publicly visible firm than when the software is associated with a smaller or younger firm.

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.[141] This is, of course, a generalization — there are open source products firms (like MontaVista and the startups discussed above) and important proprietary services firms (like EDS). 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 venture capital startup model works much better for products firms than it does for services firms,[142] so there will be relatively few startup services firms.[143] In particular, there is good reason to think that the property rights granted by patents will be uniquely valuable to

141. *See supra* text accompanying note 133.

142. CUSUMANO, *supra* note 8.

143. Mann, *supra* note 1, at 976 n.80; *see* Mann & Sager, *supra* note 1, at 26 ("[T]he great majority of venture-backed software firms are closer to the product end of the product/service continuum.").

firms attempting to progress successfully through the venture capital cycle.[144]

The comparative advantage continues throughout the business cycle. Just as the product model is better suited to the venture-backed financing common for startups, large established firms will have an advantage in the service sector. First, large, established 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. Second, as I heard repeatedly in interviews, there are considerable economies of scale in providing the 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 infrastructure than a larger company with dozens (or hundreds) of customers.

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.[145] 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. Red Hat was profitable for the first time in 2004,[146] presumably due to selling subscriptions to RHEL.[147]

---

144. Mann, *supra* note 1, at 976; *see* Mann & Sager, *supra* note 1, at 28 ("[T]here are strongly significant correlations between variables of patenting . . . and various proxies for strong performance . . . ."); *see also* John R. Allison et al., *Patents and Business Models for Software Firms* (Univ. of Tex. Sch. of Law, Law and Economics Research Paper No. 77, 2006), *available at* http://www.utexas.edu/law/faculty/rmann/info/Data/PatentsandBusinessModelsPaper.pdf (presenting empirical evidence that products firms more commonly use patents than services firms).

145. *See* Dwight Johnson, *Venture Capital Invested in Red Hat*, LINUX J., Dec. 1, 1998, http://www.linuxjournal.com/article/3171; Stephen Shankland, *Red Hat Shares Triple in IPO*, CNET NEWS.COM, Aug. 11, 1999, http://news.com.com/2100-1001-229679.html.

146. RED HAT, INC., FORM 10-K, at 18 (2005), *available at* http://www.sec.gov/Archives/edgar/data/1087423/000119312505108884/d10k.htm (reporting a profit for Red Hat in 2004, but losses in 2003 and before).

147. Subscription sales carry high profit margins in the range of eighty to ninety percent. Services sales carry much lower margins, in the range of forty to fifty percent. 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 (ninety percent versus fifty percent 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. *See* REDHAT, INC., 2004 ANNUAL

More generally, a property rights system favors new entrants because large firms can use other tools related to their market power to continue to grow (e.g., leveraging products against other products, leveraging services against products, marketing advantages). Small firms have nowhere to turn except property rights.[148] It is 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 to develop the kind of brand identification and market power that would allow it to compete against large incumbents.[149] In substance, as Figure 1 suggests, this is a basic distinction in the types of appropriation mechanisms that are useful for different types of firms.

Figure 1: Appropriation Mechanisms

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

From this perspective, open source, in the sectors where it succeeds, removes from the market firms that are developing discrete products from which they wish to get revenues.[150] So open source, viewed more fully, is highly interconnected to proprietary property rights, and it potentially could support a substantial shift in the distribution of innovation in the industry. Thus, open source, and not patents, potentially poses the largest threat to the "polyarchic" decision structure, that is, the multiplicity of small software firms under which the software industry has flourished for the last decade.[151]

The legal dispute over Linux plainly has the potential to disrupt the distribution of Linux-related products and services.[152] Open

REPORT (2005), *available at* http://investors.redhat.com/phoenix.zhtml?c=67156&p=irol-reportsannual.

148. *See supra* text accompanying note 133.

149. Interviews with Software Executives, *supra* note 5.

150. On the other hand, there are also reasons why open source aids small, proprietary startup firms. Open source software can be a useful input for these firms by decreasing the costs of completing a marketable product and helping firms focus their development expenditures on the portions of their products that are uniquely differentiating. Those firms, of course, depend directly on the property rights in the products.

151. *See generally* Wu, *supra* note 132, at 126 (discussing the distinction between hierarchical and polyarchical systems).

152. 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

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, the industry is 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 proprietary equilibrium that has prevailed until now.[153] 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.[154] Does it create a binding contract?[155] Will it be enforced as written? Will anyone who distributes open source software forever be barred from enforcing property rights? Will large patentees such as IBM use their patents 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 concerns the significance of the various open source development communities. If large firms take over much of 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 single-firm models.[156] His article raises two related points.

First, in the current environment, there is some reason to be concerned about the stability of the existing licenses, and in particular, the

---

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*, BUSINESSWEEK ONLINE, Feb. 7, 2005, 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.

153. *See* Mann, *supra* note 1, at 990–92.

154. *Cf.* Landgericht [LG] München [District Court of Munich] May 19, 2004, No. 21 O 6123/04 (F.R.G.), http://www.jbb.de/urteil_lg_muenchen_gpl.pdf (holding GPL enforceable), *translated in* http://www.jbb.de/judgment_dc_munich_gpl.pdf.

155. *See* ROSEN, *supra* note 3, at 65–66 (arguing that the GPL is not a contract because it lacks the requisite elements, and that the GPL is best treated as a license).

156. Richard A. Epstein, *Why Open Source Is Unsustainable*, FT ONLINE, Oct. 21, 2004, http://www.ft.com/cms/s/78d9812a-2386-11d9-aee5-00000e2511c8.html.

stability of communities built on reciprocal licenses like the GPL.[157] As discussed above, reciprocal licenses like the GPL 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.[158] The interviews I conducted suggest that sophisticated developers can use techniques to write programs that are adequately functional and yet technically separated from the Linux kernel as necessary to avoid "infection" by the GPL license.[159] 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 — increasing use of the GPL might give larger companies a relative advantage in working on the fringes of GPL projects.

Second, the type of license will likely have some effect on the type of software firm that can effectively use the project. For example, it is widely recognized that a more lenient license permits *more* third-party development.[160] Previous scholars have not focused, however, on the likely effects that differing licenses have on third-party development. A strong reciprocal license (like the GPL) does not impede a services firm, which should be relatively agnostic about the commoditization of the software that its customers buy.[161] Indeed, services firms should prefer Linux, given that, without formal support structures of its own, Linux leaves the area of customer support services open to third-party firms. Thus, for example, firms like 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, such as Apache or BSD. Thus, small firms like Covalent and Gluecode develop proprietary software that is designed to operate with Apache software. 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

157. A substantial part of the concern regarding the GPL relates not to its individual substantive terms but rather to its overall ambiguity. *E.g.*, Robert W. Gomulkiewicz, *De-Bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75, 83–92 (2002) (noting several such ambiguities). Whether that can be solved by reversioning is yet another difficult question for the communities that rely on that license.

158. *See supra* text accompanying note 65.

159. Interviews with Software Executives, *supra* note 5.

160. *See, e.g.*, WEBER, *supra* note 3, at 181 (explaining the various licenses and their potential implications).

161. *See* CUSUMANO, *supra* note 8.

program, discussed above, which is built on and interacts directly with the Apache HTTP Server. In addition, Apple's widely acclaimed new operating system Mac OS X rests on top of a BSD-licensed operating system (FreeBSD 3.2). Apple has layered its popular graphical user interface ("GUI") onto the UNIX-style open source operating system. Executives whom I interviewed 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 these concerns. However, as Epstein notes, the lack of a single control point in the decentralized open source development model makes a substantial shift in direction more difficult than it is for a proprietary firm.[162] The open source community is apparently aware of the problem, as it enters a period of reversioning of the GPL that would cover subsequent distributions of Linux and other open source software.[163] To date, however, there is little reason for optimism that the Free Software Foundation will promulgate a revised version of the GPL that responds to those concerns. Mozilla was unable to pass through reversioning successfully.[164] Thus, the important question for open source communities is whether they can develop the institutional structures to modify the contracts successfully or whether they will be forced to start over periodically (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 content developers that they will *not* need to start over simply to resolve licensing problems.

Beyond those short term problems lurk longer-term problems. Academics have long pondered the motivations behind the large-scale voluntary efforts of open source contributors to develop commercially valuable software.[165] A sophisticated literature offers a variety of reasons why individuals might make such contributions.[166] As commer-

---

162. *See* Epstein, *supra* note 156.

163. *See generally supra* note 118 and accompanying text.

164. Interviews with Software Executives, *supra* note 5.

165. *See, e.g.*, Benkler, *supra* note 46, at 411 n.83 (observing that "incentive-based contracts . . . may undermine voluntary cooperation").

166. *See, e.g.*, Dan M. Kahan, *The Logic of Reciprocity: Trust, Collective Action, and Law*, 102 MICH. L. REV. 71 (2003).

cialization proceeds, however, firms will more frequently justify their contributions by reference to the kinds of value chain motivations discussed above.[167] There is a risk that long-term shifts in market structure will cause such individuals' 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). For example, Dell recently withdrew from OSDL, apparently concluding that the markets in which it could exploit its core capabilities were not sufficiently related to Linux products to justify continued contributions.[168] This is a classic free-rider problem: if some of the contributors are profiting from value chain investments in Linux and others are not, those who are not profiting may 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.

Another cause for concern is 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 for all.[169]

The first step along this spectrum is the proprietary development system. This is best exemplified by Apple's personal 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. That model has allowed Apple to develop products that many users regard as the ultimate in functionality and ease of integration, though it has caused drawbacks by limiting the size of the development community that produces third-party applications for those products.[170]

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

---

167. *See supra* text accompanying notes 92–99.

168. Interviews with Software Executives, *supra* note 5.

169. *Cf.* Saul Levmore, *Property's Uneasy Path and Expanding Future*, 70 U. CHI. L. REV. 181, 181–89 (2003) (observing a fundamental disagreement over conceptions of property).

170. Lichtman, *supra* note 30, at 616.

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 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.[171]

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 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 result is quite similar to the modern commercial open source community. As discussed above, the patent licenses typically offered by open source contributors are 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) by the Linux community governed by the GPL.[172] The business case for this community is one of openly collaborative development. As suggested above, these collaborative models offer benefits from both the cost savings and the technological gains of collaborative rather than one-shop development.[173]

Major market players are constantly developing new consortia, reflecting different structures for investment in and access to technology tailored to different user markets. At the simplest 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 critical to market penetration. On 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

171. Interviews with Software Executives, *supra* note 5.

172. Given the implied nature of the GPL patent license, there is a great deal of doubt as to exactly how free users are to take and modify Linux. In practice, however, contributors with patent rights are unlikely to enforce them against users or modifiers of Linux. Interviews with Software Executives, *supra* note 5.

173. CHESBROUGH, *supra* note 94, at 49–51; *see supra* text accompanying note 94.

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, such as 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

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.[174] Others have begun to offer insurance policies.[175] Similarly, one of the distinguishing features of the subscription models for open source products is contractual protection of customers from IP claims related to the open source product.[176] As discussed above, still others are promising *not* to enforce existing property rights or (at least implicitly) promising *to* enforce existing property rights against those who threaten the movement.

Still, most agree that it is necessary to acquire more patents to remain competitive in the industry. The major corporate members of the OSDL continue to make heavy investments in patented technology: IBM, HP, and Intel have each been obtaining more than one thousand patents per year. Similarly, pure open source firms increasingly are acquiring their own patents, primarily to protect themselves from the threat of litigation.[177] 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. There is no reason to think that an increase — or decrease — in collaborative development in the software

---

174. Microsoft offers broad protection against infringement claims based on its products. Stacey Higginbotham, *How Open? That's the Big Patent Question*, DEAL.COM, Sept. 25, 2005, http://news.com.com/2100-1014_3-5877028.html. HP's indemnification plan protects against SCO attacks. Similarly, Red Hat offers a warranty to replace any infringing code. Novell offers legal indemnification against copyright infringement claims brought against Linux server software customers. Stephen Shankland, *Novell Offers Legal Protection for Linux*, CNET NEWS.COM, Jan. 13, 2004, http://news.com.com/2100-7344_3-5139632.html.

175. This is the business model of Open Source Risk Management, for whom Lloyd's of London underwrites insurance against claims of copyright or patent infringement through the use of Linux. R.J. Kiln & Co., Ltd., Open Source Compliance Representation and Warranty Insurance (2006), http://www.osriskmanagement.com/KilnOpenSourceComplianceInsurance.pdf.

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

177. *See, e.g.*, Joe Brockmeier, *Red Hat Acquiring Netscape Enterprise Solutions Software*, LWN.NET, Oct. 6, 2004, http://lwn.net/Articles/104412/.

industry will have a substantial effect on the propensity of firms outside the industry to obtain patents.

More fundamentally, it is not clear that corporate participants like the OSDL would contribute to open source development without the internalization of research and development benefits that patents facilitate. If much of the participation of proprietary firms in open source development is motivated by "value chain" returns, 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 other patents to protect its proprietary products like WebSphere that build on open source projects. 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 is the significance of the threat of patent infringement litigation. A prominent, though self-interested, study by Open Source Risk Management ("OSRM"), for example, concluded that the Linux kernel infringes 283 currently issued patents.[178] One highly informed executive suggested that there are about two hundred crucial patents, access to which is necessary to distribute a modern operating system.[179] 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. The mere threat has had numerous effects, ranging from the contractual assurances discussed above to some difficult-to-gauge disruption of Linux adoption. A number of events suggest that large firms not participating in open source value chains might exploit their portfolio against firms that are participating in those value chains.[180] It is not clear, however, that the enforcement risk in fact is substantial.

First and most important, the risk remains largely hypothetical. There has not yet been a patent infringement suit challenging the use or development of Linux or any of the major open source programs. Indeed, it appears that 2006 has brought some of the first patent in-

178. Sean Michael Kerner, *Linux's Patent Risk*, INTERNETNEWS, Aug. 2, 2004, http://www.internetnews.com/dev-news/article.php/3389071.

179. Interviews with Software Executives, *supra* note 5.

180. A salient event here 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, an open source suite based on code donated by Sun.

fringement lawsuits of any type against open source programs, and it is not yet clear that any of those lawsuits will be significant.[181]

Second, the risk of litigation is easily overstated. This fear stems from three factors. First, there are thousands of software patents that cover open source software programs — perhaps ten thousand patents that cover operating systems alone, so the risk of infringement is high. Second, open source developers often do not obtain patents, suggesting that they are unconcerned with potential infringement. Third, infringement by open source products is more detectable than infringement by proprietary products because the source code will display the algorithm of the software.[182]

The first of these three factors is undermined by the limited usefulness of patents as a tool for appropriating innovations in software. 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 the patent.[183] This is true for a variety of reasons, the most important of which is that the pattern of software innovation provides multiple paths to most design problems.

The first and second factors are undermined by the fact that at least some of the largest firms in the industry have patent-reviewing practices that avoid and minimize interference with 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 patent claims that might read on products of the firm.[184] Lawyers in these programs commonly discover such patents, and the firms respond promptly to alleviate the problem. Depending on the seriousness of the problem, a range of obvious responses ap-

181. One public example involves a suit by a software startup called FireStar against Red Hat, challenging Red Hat's use of certain database technology that Red Hat recently acquired when it purchased JBoss. Bruce Perens, *The Monster Arrives: Software Patent Lawsuits Against Open Source Developers*, TECHNOCRAT.NET, June 30, 2006, http://technocrat.net/d/2006/6/30/5032. Another is a claim by Michael Katzer and KAM against Bob Jacobsen, who has developed a successful open-source program for controlling model railroads. *Id.* The most significant threat appears to be the initiation of litigation by Blackboard against other e-learning technologies, which is widely viewed as a threat to a number of related open source projects. *See* Posting of John Ottaviani to Eric Goldman: Technology & Marketing Law Blog, http://blog.ericgoldman.org/archives/2006/08/blackboard_pate.htm (Aug. 8, 2006, 20:14 EST).

Another significant event on that front is Kodak's recent successful lawsuit against Sun claiming that Sun's Java technology infringed 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 position in its major existing market, Kodak has almost as little to lose as SCO by aggressive litigation. Interviews with Software Executives, *supra* note 5.

182. *See* Zittrain, *supra* note 96, at 285–86.

183. Mann, *supra* note 1, at 978–80.

184. Interviews with Software Executives, *supra* note 5.

pears: 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.

All of these responses are common in the industry, individually and in combination. The OSDL engages in similar activity with respect to commercially significant open source programs like Linux and Apache. Collectively, these two criticisms undermine the first and second factors 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.

The second factor about developers being unconcerned with patent infringement is becoming less true even in the open source community. Whatever might be true for the "hard-core" portions of the community associated with the Free Software Foundation, "disdain" for patents is no longer a catechism 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 work around the third-party patent.

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 risk of litigation 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[185] — or if a party such as a troll that is outside the existing industry holds the patent. It certainly is plausible that a troll could obtain a "nuclear bomb" patent that would read onto major commercial software platforms. It is hard to say, 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.

185. *See, e.g.*, Mann, *supra* note 1, at 1005 (providing reasons why IBM does not stringently enforce its patents).

V. CONCLUSION

Some academics see the open source movement as a victim of an excessive intellectual property system and fear that it cannot coexist 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 research and development through IP investments and predict the abandonment of IP-centric development models.

This Article 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, mirroring 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 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 coexist, in a world in which property rights will continue to matter. In addition, if they continue to coexist, 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 open source's creators.