ARTICLE

PROMOTING INNOVATION IN THE SOFTWARE INDUSTRY: A FIRST PRINCIPLES APPROACH TO INTELLECTUAL PROPERTY REFORM

BRUCE ABRAMSON[*]

---

[*] Ph.D., (Computer Science) Columbia, 1987; J.D., Georgetown, 2000. Dr. Abramson is a Principal of Charles River Associates and an Adjunct Professor of Engineering and Public Policy at Carnegie Mellon University. He may be contacted at CRA, 1201 F St. NW, Suite 700, Washington, DC 20004-1204, (202) 362-3181, babramson@crai.com, fax (202) 662-3910. The author would like to thank Julie Cohen and Mark Lemley for comments on earlier versions of this article. The opinions expressed in this article are the author's alone, and do not necessarily reflect the views of either Charles River Associates or Carnegie Mellon University.

I. ABSTRACT

The Intellectual Property (IP) clause of the U.S. Constitution encourages Congress to promote the development of art and science. The Constitution also instructs Congress to achieve this goal by allowing authors and inventors to retain exclusive rights to their innovations and thus to profit from their commercialization. No further details are provided. These instructions may be viewed as the "first principles" of the U.S. IP system: harnessing the profit motive to promote artistic and scientific progress.

Throughout most of American history, two primary categories of IP rights have sufficed to promote this progress: patents and copyrights. The promotion of innovative software presented the IP system with a unique set of challenges. Software shares some characteristics with the innovations generally protected by patents, and others with innovations generally protected by copyright. It does not fit neatly into either category. Nevertheless, a unique combination of patent, copyright, and trade secret law—complete with a set of *sui generis* exceptions—has emerged to protect software.

This article argues that the existing system of software rights is inconsistent with the first principles of our IP system. It shows how the combination of protections now available to software developers provides a sub-optimal incentive structure for innovation. Under the current regime, knowledge is hoarded rather than shared. Products mature more slowly than they might under a more appropriately tailored regime, and firms may be rewarded for anticompetitive behavior. This article does not contend that the existing IP regime has failed to generate a vibrant software industry. It does, however, show how different protective regimes could have led (and could still lead) to a more rapid dissemination of knowledge, to more intense commercial competition, and to superior software products.

The article demonstrates these points within an analytic framework designed to highlight a general point. Software may be the first industry for which neither patents nor copyrights are a natural fit; it is unlikely to be the last. A firm grounding in first principles is necessary to design appropriate protection for all such industries.

I. INTRODUCTION

Quick. Is Microsoft Windows like Joyce's *Ulysses*, Edison's light bulb, or the formula for Coca-Cola?

The answer is almost certainly "none of the above." For a variety of reasons, however, "none of the above" is not an acceptable answer—at least not within the confines of intellectual property ("IP") law. From an IP perspective, creative innovations are divided into a small number of neat categories. Textual and artistic works—like *Ulysses*—are protected by copyright.[1] Functional inventions—like the light bulb—are granted patents.[2] Recipes and formulas—such as the one for Coca-Cola—may be maintained as trade secrets[3] and protected against espionage, but they become ineligible for legal protection once someone other than their creator has discovered (or dissected) them.[4] Windows, a well-known example of computer software, violates these categorical distinctions. All computer programs are textual works designed to be functional,[5] thereby suggesting a need for both copyright and patent protection. Because many programs are also maintained as proprietary corporate secrets, trade secret law has also played an important role in the development of the software industry.[6]

The schizophrenic nature of software emerges (at least in part) from its dual identity as source code and object code. Programs can be viewed as textual source code that is comprehensible to trained programmers, or as functionally equivalent compiled object code that only computers can understand. Since many software manufacturers choose to keep their source code secret while distributing copies of their object code, their interests focus on maintaining strict proprietorship of their source code and on reducing instances of improper copying and circulation of their object code. These concerns implicate multiple branches of IP law.

---

[1] *See* 17 U.S.C. § 102(a)(1) (2000).

[2] *See* 35 U.S.C. § 101 (2000).

[3] *See* ROGER M. MILGRAM, MILGRAM ON TRADE SECRETS § 1.01 (2001) (A trade secret is "a formula, pattern, device or compilation of information which is used in one's business, and which give him an opportunity to obtain an advantage over competition, who do not know it or use it . . . The subject matter of a trade secret must be a secret."); Coca-Cola Bottling Co. v. Coca-Cola Co., 107 F.R.D. 288, 289 (D. Del. 1985) ("The complete formula for Coca-Cola is one of the best-kept trade secrets in the world.").

[4] *See id*.

[5] *See* 17 U.S.C. § 101 ("A 'computer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.").

[6] There is one additional category of IP protection: trademark. While software trademarks raise a variety of interesting issues, they are tangential to this article. Patents, copyrights, and trade secrets all deal with innovations in which society would like to motivate investment. Trademarks arise from a distinct set of concerns, namely consumer protection. As a result, software trademarks require a different analysis that will not be considered here.

In short, Windows is like—and unlike—all of *Ulysses*, the light bulb, and Coke. This situation posed a quandary for the policy makers who first had to decide how software should be categorized and protected. That quandary has since become a morass. In contemporary America (as well as in most of the rest of the developed world) some software is protected by patents, some by copyrights, and some by both. There are surprisingly few bright lines dividing patentable from unpatentable software. Most commercial software is protected against piracy by its patents and/or its copyrights, while trade secret law provides an added layer of protection for source code. Traditional means of circumventing trade secret protection, such as various forms of reverse engineering,[7] raise a number of unique legal issues when applied to software. The courts and Congress have both attempted to address these issues, but the line dividing permissible from impermissible circumvention remains somewhat blurred.[8]

The challenge inherent in shoehorning software into one of these few categories is representative of a broader problem. Innovations in agrarian and industrial societies may have lent themselves to a meaningful bipartition. Copyrights provided artists with one class of commercial opportunities; patents provided inventors with another.[9] These opportunities were believed to motivate appropriate levels of innovation within both groups of innovators. Our contemporary technological society contains many different types of innovation—in areas as diverse as information and genetics—and innovators whose expected financial returns accrue in various ways and at various speeds. Our continued reliance on only two forms of protection is likely to be under-rewarding innovation in some fields (and thus retarding progress), and over-rewarding it in others (and thus needlessly elevating the societal cost of progress).

This article addresses both this broad theoretical concern and the more focused challenge of protecting software. It proposes a general analytic

---

[7] *See* DONALD S. CHISUM & MICHAEL A. JACOBS, UNDERSTANDING INTELLECTUAL PROPERTY LAW § 3E[3] (1992) (Reverse engineering involves "starting with the known product and working backward to find the method by which it was developed."), citations omitted, [hereinafter UNDERSTANDING INTELLECTUAL PROPERTY LAW].

[8] This blurring, and the judicial and legislative attempts to clarify it, are discussed in the text *infra* § VI.B.1.

[9] *See* ROBERT P. MERGES ET AL., INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE 23-27 (2d ed. 2000). A patent grants exclusive rights to make, use and sell an invention for up to 20 years. *See id.* at 23. "The patent grant is nearly absolute, barring even those who independently develop the invention from practicing its art . . . . In general copyrights are easier to secure and last substantially longer than patents, although the scope of protection afforded copyrights is narrower and less absolute than that given to patents." *Id.* In theory the justification for granting these rights is based on the notion that granting legal protection for "ideas" creates an incentive for people to create or invent. *See id.* at 12. The differences between patents and copyrights thus implicitly create different commercial opportunities.

framework for assessing the propriety of a protective regime *as applied to a specific industry*, and then uses the software industry as the first case study of this framework's applicability. The analysis demonstrates that although the existing bifurcated IP regime has generated a vibrant software industry, an alternative set of protective rights could probably have done better. In particular, a system of software rights designed with an eye on some of the industry's unique characteristics would have been likely to motivate innovation comparable to the current system, to provide society at large with greater access to scientific and technological advances, and to bestow additional benefits on consumers, all while reducing the costs borne by society.[10]

This article contains eight sections. Section 2 reviews some background information that motivates the reconsideration of IP rights in the software industry. It also introduces an influential proposal for reforming software protection first suggested in 1994 by a team of technologists and IP scholars.[11] This proposal is used throughout the article as a foil to the current regime. The contrast between these regimes helps illustrate the ways in which different types of protection could have led (and could still lead) to different configurations of the software industry.

The next three sections develop the article's underlying theory. Section 3 presents a statement of first principles: IP rights exist solely to motivate innovation.[12] Section 4 discusses the basic economics of incentives, investments, and innovation necessary to evaluate tradeoffs between a set of IP rights and the innovations that it is expected to motivate. This discussion leads to section 5's four-stage framework for the analysis of a specific industry: (i) Characterize the industry; (ii) Define the protective regime; (iii) Calculate the potential return on private investment; and (iv) Consider the societal costs and benefits.

Section 6 sets the software industry in section 5's framework. Section 6.1 provides the first stage—a description of the software industry. Section 6.2

---

[10] One caveat is required before proceeding with this demonstration. This article is about analysis and policy. It shows that software rights derived from the first principles of the IP system would look very different from those that currently exist. The implications of this analysis include prescriptions for *future* IP rights. Nothing in this discussion should be construed as disparaging existing rights or encouraging the infringement of those rights. The software industry in its current form is an important contributor to the world economy. The rights that underpin it—improvidently granted or not—must continue to be respected unless and until they are changed.

[11] *See* Pamela Samuelson, Randall Davis, Mitchell D. Kapor & J.H. Reichman, *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2422-25 (1994) (advocating short-term anti-cloning protection for software as a means of protecting new software innovations long enough to allow development of a market for them) [hereinafter *Manifesto*].

[12] *See* U.S. CONST. art. I, § 8, cl. 8 (giving Congress the power "To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries.").

contains the second stage description of two protective regimes: the current one and the reform proposal mentioned above. Section 6.3 begins the cost/benefit discussion central to the third and fourth analytic stages. Section 7 summarizes the comparative analysis, and demonstrates the likely net superiority of industry-tailored rights. Section 8 offers some concluding thoughts.

II. BACKGROUND

This article interleaves discussions of a general phenomenon—the potential shortcomings of existing legal doctrine in motivating innovators—and the specific manifestation of that phenomenon in the software industry. The selection of the software industry as a concrete illustration was hardly arbitrary. It was chosen because it is among the largest, the most mature, and the best studied of the post-industrial age industries. The challenge of protecting software innovation has blurred the previously bright lines dividing the realms of patent and copyright. This section reviews many of the issues raised by courts and commentators as they recognized this phenomenon. It provides background on two key issues: the challenge inherent in deciding how to reward innovative software development (section 2.1), and some proposed approaches towards meeting that challenge (section 2.2).

A. *Patents, Copyrights, and Antitrust*

Perhaps the first question that commercially focused software developers (or more likely, their legal counsel) asked themselves was whether their innovations would be protected by patent or by copyright. While the current answer is a clear "it depends," this question had no easy answer during the early years of the software industry. On the copyright front, questions lingered at least through the 1970s.[13] In 1974—already the third decade of software's existence—Congress established the National Commission on New Technological Uses of Copyrighted Works (CONTU).[14] Four years later, CONTU recommended extending copyright law to cover software in line with prevailing industry expectation and practice.[15] Although the commission's conclusions were not unanimous,[16] Congress enacted CONTU's recommendations into law in 1980, and software became explicitly copyrightable.[17]

---

[13] *See* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663, 693-94 (1984).

[14] Act of December 31, 1974, Pub. L. No. 93-573, 88 Stat. 1873 (1974) (codified as amended at 17 U.S.C. § 701 (2000)).

[15] *See* Samuelson, *supra* note 13, at 666.

[16] *See id.* at 698-99.

[17] Pub. L. No. 96-517, 94 Stat. 3015 (1980) (codified as amended at 17 U.S.C. § 117 (2000)). For a brief overview of CONTU, and of Commissioner Hersey's dissent, *see* MERGES ET AL., *supra* note 9, at 911-13.

The path to patent protection was even more tortuous. Long-standing patent principles prohibit patenting an idea or a mathematical formula.[18] Computer programs were viewed as textual representations of mathematical algorithms and thus potentially appropriate subjects for copyright protection—but not for patents.[19] As a result, the Patent and Trademark Office (PTO) consistently refused to award patents to software developers, and a combination of copyright and trade secret law came to define the industry throughout the 1950's and 1960's.[20] The first serious doubts about this paradigm emerged in 1972, when the Supreme Court accepted the general principle of a software patent (although it refused to grant one at that point).[21] At the risk of oversimplification, the Court introduced a hypothetical split between simple algorithms (unpatentable) and algorithms embedded in specific applications (potentially patentable subject matter).[22] The Court refined this split and found a patentable embedded algorithm in 1981.[23]

The pendulum has since swung towards increasingly lower barriers for software patents, as applied to both algorithms and business methods. In the midst of the 1999 holiday shopping season, for example, a District Court ruling granted a preliminary injunction allowing Amazon.com to enforce its patent on the "one-click" method of ordering goods over the Internet, despite Barnesandnoble.com's contention that the patent was invalid because the PTO had paid insufficient attention to the prior art.[24] Although the Federal Circuit subsequently vacated that ruling and remanded the case for further proceedings,[25] the fact remains that the PTO granted the patent, and the courts allowed it to affect Internet commerce during a key fourteen-month period. This appellate ruling may (or may not) mark yet another turnaround and an acknowledgment that the PTO's standards for business process patents had fallen too low—an issue that the PTO itself had reportedly been reassessing.[26]

The challenge of categorizing software thus posed some serious definitional problems that worked their way to the highest levels of the political and legal

---

[18] *See* DONALD S. CHISUM, CHISUM ON PATENTS § 1.01, § 1.03[2][d] (citing Gottschalk v. Benson, 409 U.S. 63 (1972)) (2001) [hereinafter CHISUM ON PATENTS].

[19] *See* Mackay Radio & Tel. Co. v. Minnesota & Ont. Paper Co., 306 U.S. 86, 94 (1939) (holding that a mathematical representation of a scientific truth is not patentable).

[20] *See* Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329, 1348 (1987) (stating that Patent and Trademark Office initially refused to patent computer programs).

[21] *See* Gottschalk v. Benson, 409 U.S. 63, 71-72 (1972).

[22] *See id.* at 65, 71.

[23] *See* Diamond v. Diehr, 450 U.S. 175, 191-93 (1981).

[24] *See* Amazon.com, Inc. v. Barnesandnoble.com, Inc., 73 F. Supp. 2d 1228, 1233, 1235 (W.D. Wash. 1999) *vacated and remanded by* 239 F.3d 1343 (Fed. Cir. 2001).

[25] *See* Amazon.com, Inc. v. Barnesandnoble.com, Inc., 239 F.3d 1343, 1347 (Fed. Cir. 2001).

[26] *See* Sabra Chartrand, *Federal Agency Rethinks Internet Patents*, N.Y. TIMES, Mar. 30, 2000, at C12.

systems—and that nevertheless continue to cause controversy. The underlying problem, however, runs deeper than ambiguity at the edge of legal doctrine. Despite the availability of software patents, most software remains protected by the aforementioned combination of copyright and trade secret law. The use of copyrights to protect the functional innovations embodied in software has allowed the rights holders to exert their rights in ways that could not have been imagined by the drafters of the original Copyright Act.[27] Many of these exertions parallel activities that have been banned either under the antitrust laws or under the patent law doctrine of patent misuse.[28] Because copyright law was not designed with functional innovations in mind, it contains no comparable traditional doctrine to address these potentially undesirable (or illicit) exertions of rights.[29] This understandable shortcoming sets the stage for an inevitable tension between copyright law and antitrust law as applied to software.[30]

---

[27] *See* Samuelson, *supra* note 13, at 705-06 (explaining that the *quid pro quo* of copyright law is intended to be legal protection in exchange for the disclosure of new ideas to the general public but that computer software released as object code readable only by machines secures the legal protection while subverting the public benefit).

[28] *See generally* MERGES ET AL., *supra* note 9, at 303-14, 1101-98; *see also* the discussion in *infra* note 48. Patent misuse is a judicially crafted doctrine that prevents patent holders from "misusing" their patent rights—typically by attempting to extend them beyond the range for which they were granted. *See* MERGES ET AL., *supra* note 9, at 303. Much of the behavior proscribed under this doctrine is similar to activities that qualify as antitrust violations. *See id.* at 307. While the original conception of patent misuse was broader in scope than the antitrust laws, it has been narrowed considerably since it was first articulated in 1917. *See id.* at 307-14. The two bodies of law are now quite similar—although patent misuse does still cover a number of practices that arise exclusively or primarily in the context of patents. *See id.*

[29] There is a nascent doctrine of copyright misuse that has been adopted by several of the Circuits. *See* Brett Frischmann & Dan Moylan, *The Evolving Common Law Doctrine of Copyright Misuse: A Unified Theory and its Application to Software*, 15 BERKELEY TECH. L. J. 865 (2000) (tracing the development of this doctrine back to its common law roots and discussing its specific, recent development within the realm of software litigation). The doctrine is discussed in greater detail in *infra* n. 185.

[30] The DC Circuit's recent response to an argument forwarded by Microsoft captures this tension well:

> The company claims an absolute and unfettered right to use its intellectual property as it wishes: '[I]f intellectual property rights have been lawfully acquired,' it says, then 'their subsequent exercise cannot give rise to antitrust liability.' That is no more correct than the proposition that use of one's personal property, such as a baseball bat, cannot give rise to tort liability. As the Federal Circuit succinctly stated: 'Intellectual property rights do not confer a privilege to violate the antitrust laws.'

United States v. Microsoft Corp., 253 F.3d 34, 63 (D.C. Cir. 2001) (citations omitted) (per curiam), *reh'g denied*, 2001 U.S. App. LEXIS 17137 (D.C. Cir. Aug. 2, 2001), *mot. denied*, 2001 U.S. App. LEXIS 18175 (D.C. Cir. Aug. 17, 2001), *cert. denied*, 2001 U.S. LEXIS 9509 (U.S. Oct. 9, 2001) [hereinafter *Microsoft*-Appeal].

To appreciate the source of this tension, it is important to recall that IP protection—of any flavor—confers a monopoly on the holder of the IP rights. This legally bestowed monopoly thus enables the rights holder to license her protected innovation and to collect royalties from authorized users. While relatively few of these "monopolies" confer any type of meaningful market power, the holders of the select few valuable IP rights can emerge as significant monopolists. These powerful rights holders may propose (and even extract) license terms that violate the antitrust laws. The law's perception of the boundary between permissible and impermissible license terms has shifted over time; terms that have been considered valid assertions of IP rights during some eras would have been viewed as misuse and/or as antitrust violations in others. [31] This movable boundary defines an inherent tension between the IP laws and the antitrust laws.[32] Prior to the advent of software, however, this tension lay almost entirely in the realm of patent law.

The courts' initial view of this boundary was highly deferential to patent rights. In a classic 1926 case, General Electric ("GE") licensed patented light bulb technology to Westinghouse under terms that required Westinghouse to adhere to GE's pricing schedule.[33] The government argued, *inter alia*, that this term constituted resale price maintenance, or vertical price fixing,[34] a practice that had already been found illegal under the antitrust laws.[35] Chief Justice Taft, writing for the Court, agreed with the government's view that GE would have been guilty of vertical price fixing had it applied comparable terms to an unpatented product.[36] Nevertheless, the Court also agreed with GE that its patent protected it from the allegation, reasoning that since GE could legally have withheld the technology from Westinghouse altogether, there was no reason to disallow this strictly less restrictive license term.[37]

While *General Electric* has never been overturned, a collection of cases narrowing its holding appeared throughout the middle decades of the twentieth century.[38] This case law asserted, for example, that "the authorized sale of a

---

[31] *See infra* notes 33-44 and accompanying text.

[32] *See* MERGES ET AL., *supra* note 9, at 1105-10.

> Traditionally, the conventional wisdom was that the antitrust laws and the intellectual property laws are in conflict . . . . Baldly stated, the conflict arises because the intellectual property laws grant 'monopolies' to inventors, while the goal of the antitrust laws is to prevent or restrict monopoly. However, scholars are increasingly taking the position that the two laws are not in conflict at all. Rather, they are complementary efforts to promote an efficient marketplace and long-run, dynamic competition through innovation.

Id. at 1105.

[33] *See* U.S. v. General Electric, 272 U.S. 476, 479 (1926).

[34] *See id.* at 479-80.

[35] *See* Dr. Miles Med. Co. v. John D. Park & Sons Co., 220 U.S. 373, 405 (1911).

[36] *See General Electric*, 272 U.S. at 486.

[37] *See id.* at 490.

[38] *See* U.S. v. United States Gypsum Co., 333 U.S. 364, 400 (1948) (patent holders

[patented] article. . .is a relinquishment of the patent monopoly"[39] and that "[t]he first vending of any article manufactured under a patent puts the article beyond the reach of the monopoly which that patent confers."[40] These mid-century cases severely narrowed the range of restrictions that a patent holder could place on her licensees and shifted the balance away from patent rights towards antitrust law.

The pendulum may have swung back in a more permissive direction since the creation of the Federal Circuit.[41] In a key 1992 ruling, the Federal Circuit reversed a summary judgment against Mallinckrodt despite some restrictive conditions that its licenses placed on the reuse of a patented medical device.[42] The court remanded the case for a new trial to determine whether the prohibition on reuse was closer to a restriction on repair (an activity that is generally permissible under the patent law) or reconstruction (an activity that is among the rights reserved by the patent-holder even in the absence of an explicit license term).[43] The Supreme Court has yet to comment on the direction taken by the Federal Circuit—a direction that appears to be rather deferential to patent rights.[44]

Software and other functional texts have allowed the IP/antitrust tension to migrate from patent law into copyright law. The government's antitrust case against Microsoft[45] is probably the clearest and most widely discussed—but

---

cannot organize the use of their patents throughout an entire industry by regulating distribution through licenses); U.S. v. Masonites Corp., 316 U.S. 265, 279 (1942) ("A patentee who employs such an agent to distribute his product certainly is not enlarging the scope of his patent privilege if it may fairly be said that that distribution is part of the patentee's own business and operates only to secure to him the reward for his invention which Congress has provided. But where he utilizes the sales organization of another business—a business with which he has no intimate relationship—quite different problems are posed since such a regimentation of a marketing system is peculiarly susceptible to the restraints of trade which the Sherman Act condemns."); U.S. v. Univis Lens Corp., 316 U.S. 241, 249 (1942) (stating that at least some patent rights are operative only through the first sale of the patented article).

[39] *Univis Lens Corp*., 316 U.S. at 249.

[40] *Id.* at 252.

[41] Congress created the Federal Circuit in 1982 with the passage of the Federal Courts Improvement Act, in part to unify patent doctrine by establishing a single court to hear all appeals in cases asserting patent claims. *See* MERGES ET AL., *supra* note 9, at 129-30.

[42] *See* Mallinckrodt, Inc. v. Medipart, Inc., 976 F.2d 700, 708-09 (Fed. Cir. 1992).

[43] *See id.* at 709.

[44] In addition to these shifts in the courts' attitudes, the enforcement agencies (*i.e.*, the Justice Department and the Federal Trade Commission) have also moved the line between rights that accompany a patent and activities that violate the antitrust laws. Their current thinking about this interface shows an increased deference to patent rights consistent with the trend in the courts. *See* U.S. DEP'T. OF JUSTICE & FTC, ANTITRUST GUIDELINES FOR THE LICENSING OF INTELLECTUAL PROPERTY, § 4 (1995).

[45] Reporting of the first round of the *Microsoft* trial was split. The findings of fact are in United States v. Microsoft Corp., 65 F. Supp. 2d 1 (D.D.C. 1999) *aff'd in part, rev'd in part,*

hardly the only—demonstration of this tension. Microsoft was found guilty of antitrust violations for a variety of negotiating tactics and licensing terms associated with Windows, a software package protected by a combination of patent, copyright, and trade secret laws.[46] A unanimous *en banc* D.C. Circuit upheld the illegality of most of these tactics; Microsoft was found to have improperly maintained its (previously earned) monopoly over operating systems for PCs based on Intel's microprocessors.[47]

The behavior described in *Microsoft* is analogous to a pattern that has long been common at the patent/antitrust interface[48] but implicates an important new twist: Many of the rights asserted were protected by copyright, rather than by patent. The novelty of this twist emerged from a confluence of events. Whereas an *operating system patent* would have conferred a fairly broad monopoly on Microsoft, the *Windows copyright* was fairly narrow.[49] Microsoft had to earn the extension of its monopoly from Windows to operating systems running on Intel-based PCs[50] by defeating all competing operating systems in the marketplace. Microsoft's monopoly was thus

---

remanded, 253 F.3d 34 (D.C. Cir. 2001), *cert. denied* 2001 U.S. LEXIS 9509 (Oct. 9, 2001) [hereinafter *Microsoft*-Facts]. The findings of law are in United States v. Microsoft Corp., 87 F. Supp. 2d 30 (D.D.C. 2000) *aff'd in part, rev'd in part, remanded*, 253 F.3d 34 (D.C. Cir. 2001), *cert. denied*, 2001 U.S. LEXIS 9509 (Oct. 9, 2001) [hereinafter *Microsoft*-Law].

[46] *See Microsoft*-Law, 87 F. Supp. 2d at 35.

[47] *See Microsoft*-Appeal, 253 F.3d at 46.

[48] In this classic pattern, a patent holder agrees to license her patent subject to a variety of conditions. Some of these conditions restrict the licensee's ability to make independent business decisions. The licensee (or, as in *Microsoft*, the government on behalf of consumers) brings suit, objecting that these restrictions limit his ability to compete, or possibly even restrict the entire competitive structure of the market. The patent holder counters that the license terms are nothing more than a legitimate exercise of her patent rights. Suits of this sort raise questions of both antitrust and patent misuse. The patent misuse doctrine was first articulated in Motion Picture Patents Co. v. Universal Film Manufacturing Co., 243 U.S. 502, 516 (1917). It has gone through several incarnations in the past eighty-plus years—some interpreting the violation broadly and some narrowly. Its relationship to antitrust law—and in particular to the antitrust violation of tying—has remained a matter of some controversy throughout all of these incarnations. Judge Posner, for example, has argued that it should be coextensive with antitrust law, and in particular the antitrust violation of tying. *See* USM Corp. v. SPS Technologies, 694 F.2d 505, 510 (7th Cir. 1982), *cert. denied*, 462 U.S. 1107 (1983). At the time of his assertion, however, there was still a body of good law suggesting that patent misuse was broader in scope than tying. Congress has since entered the fray to narrow at least some of this excess scope with the Patent Misuse Reform Act (PMRA) of 1988. *See* Pub. L. No. 100-703, § 201, 102 Stat. 4674 (codified as amended at 35 U.S.C. § 271(d) (2000).

[49] The concepts of protective breadth and depth are discussed in detail in *infra* § 4.3 of the text.

[50] The market for such Intel-based PCs is that which Microsoft monopolized. *See Microsoft*-Appeal, 253 F.3d at 45.

partially granted and partially earned—and as the court noted, maintained at least in part through illegal means.[51]

Courts have developed a variety of responses, not all of which are mutually consistent, to this newfound ability to leverage copyrights in an anticompetitive manner. Some have crafted a decompilation exception to copyright law, creating some ambiguity around the issue of reverse engineering.[52] At least part of this exception has been codified in the Digital Millennium Copyright Act ("DMCA"),[53] thereby removing some but not all of the ambiguity.[54] Others have begun to develop a new doctrine of "copyright misuse" that addresses the newly relevant boundaries between *functional* copyrights and antitrust law.[55] Either way, both Congress and the courts have recognized that some type of reform—or at the very least, flexibility—is required.

B. *Proposed Reforms*

Commentators drawn from the legal and the technological communities have also voiced their opinions about the challenges posed by software that were simply not implicated in the protection of more traditional innovative products. In the early days of software development, copyright protection appeared to be a much better, albeit imperfect, fit than patent protection.[56] As the industry matured, many of the problems inherent in this policy choice became evident. By 1987, these difficulties were sufficiently evident for Menell to present a cogent (and in many ways prescient) analysis of the impropriety of copyright protection for software.[57]

In the ensuing years, a number of proposals have been forwarded. These proposals may be broken into two broad classes: radical reforms pushing for a new *sui generis* form of protection for software, and conservative reforms pushing for increased flexibility in the application of existing legal doctrines to software. Perhaps the most significant of the radical reforms was presented as part of a symposium held at Columbia University in 1994. The key article,

---

[51] *See id*.

[52] *E.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); Atari Games Corp. v. Nintendo of America, Inc. 975 F.2d 832 (Fed. Cir. 1992).

[53] *See* Pub. L. No. 105-304, 112 Stat. 2877 (1998) (codified as amended at 17 U.S.C. § 1201 (2000)).

[54] For a more complete discussion of the decompilation exception *see infra* note 219 and accompanying text.

[55] Both of these responses are discussed in detail elsewhere in this article. Reverse engineering is discussed in *infra* note 219. The copyright misuse doctrine is discussed in *infra* note 185 and accompanying text.

[56] *See* Menell, *supra* note 20, at 1347-51 (discussing the development of patent protection for software and the shortcomings of this form of IP protection for software); Samuelson, *supra* note 13, at 692-94 (discussing early application of copyright to software).

[57] *See* Menell, *supra* note 20, at 1359-64.

commonly known as "The Manifesto,"[58] set out to "contribute a basic framework for constructing a new form of legal protection for program innovations."[59]

The Manifesto's analysis began by identifying several key characteristics of computer programs: (i) They behave, (ii) they are constructed from text, and (iii) they evolve incrementally from one generation to the next.[60] Each of these features provides insight into the types of innovation that must be promoted to ensure the development of a healthy software industry. These features also imply the inadequacy of existing IP rights. Traditional copyrights, for example, protect expressions of ideas, not ideas themselves.[61] To a novelist or an artist, this protection is meaningful. To a computer programmer, it is not. While there may be a pride of authorship that accompanies a well-written piece of computer code, a program's *raison-d'être* is its behavior. Programs are written to perform specific tasks. Software innovators are driven to automate increasingly sophisticated behavior. An IP right that provides no protection to the newly discovered behavior, and that allows any other programmer to mimic the behavior simply by writing new code from scratch, fails to protect the valuable item that motivated the innovation. At the same time, the incremental nature of software development suggests that most programs would fail either the novelty or non-obviousness tests of patent law.[62] Most computer programs, including most important software innovations, represent only slight movements in the state of the art.[63] The first key to the Manifesto's analysis was thus a demonstration of the inadequacy of traditional IP rights, and a conclusion that the use of traditional legal regimes to protect software innovations would lead to inevitable cycles of under- and over-protection.[64]

The Manifesto then developed a set of abstract principles and goals for a market-oriented approach to the protection of software innovations and sketched a proposal for a new regime that paralleled many important motivational components of patent law.[65] Under this proposal, software developers would be able to register their programs to obtain protection for the

---

[58] *Manifesto, supra* note 11.

[59] *Id.* at 2315.

[60] *See id.* at 2315-16.

[61] WILLIAM F. PATRY, COPYRIGHT LAW AND PRACTICE, VOL. I, at 312 (1994) (citing 17 U.S.C. § 102(b) (1978)).

[62] "The novelty requirement lies at the heart of the patent system." CHISUM ON PATENTS, *supra* note 18, at § 3.01. The novelty requirement ensures that patentable inventions are "new," a somewhat subjective term defined formally by statute. *See id.* Novelty is closely related to the requirement of nonobviousness; an invention that is new enough to qualify as novel must meet the additional requirement that it not be obvious to one with ordinary skill in the art implicated by the invention. *See id.*

[63] *See Manifesto, supra* note 11, at 2330-31.

[64] *See id.* at 2356.

[65] *See id.* at 2405-13 (listing fifteen goals and principles).

innovative behavior that they embodied.[66] In exchange, the developers would be required to disclose their programs—including their source code.[67] In this way, at least, the Manifesto's proposal appears to mimic the patent system. The proposal differs from patent law, however, in a few key respects. In particular, the proposed protection would be easier to earn (*i.e.*, it would have to meet lower thresholds of novelty and nonobviousness than a standard patent application), and it would expire much more quickly.[68]

The Manifesto's proposal was but one of several radical responses to the challenges posed by software. Other revolutionary proposals have included the creation of various hybrid regimes and/or other types of *sui generis* forms of protection.[69] Proponents of even radical reform, however, have recognized that patent law and copyright law provide a wealth of information about legal mechanisms that have succeeded and failed in a variety of different settings. Advocates of conservative reform have recognized that the historical development of legal doctrines within patent and copyright law, as well as the occasional cross-fertilization that results in nascent doctrines like copyright misuse, provide the building blocks from which any reform proposal should be drawn. In the specific context of software protection, the nature of the industry and several decades of experience should provide even further guidance.

These concerns played an important role in *The Digital Dilemma*, a recent a study commissioned by the National Research Council.[70] The study concluded that, although there was a clear need for new forms of IP protection, legislation at this point would be premature.[71] Its ultimate recommendation was that Congress observe the various "experiments" already underway in business (*i.e.*, relying on combinations of contracts and technology to protect software) and in the courts (*i.e.*, the development of new, relevant IP doctrines and/or exceptions) to gain useful data in assessing appropriate reform.[72]

---

[66] *See id.* at 2417-18.

[67] *See Manifesto, supra* note 11, at 2417-18. For a discussion of source code and object code in software programming, *see* discussion in *infra* § VI.B.

[68] Some of the Manifesto's key points were recognized by a source as unlikely as the holder of the business method patent of one-click Internet ordering, namely Amazon.com Chairman Jeff Bezos, who suggested that a three-to-five year lifetime would be adequate for software patents. *See* Matt Richtel, *Chairman of Amazon Urges Reduction of Patent Terms*, N.Y. TIMES, Mar. 11, 2000, at C4. Bezos's comments were made shortly after a District Court upheld the patent (a decision that was later reversed and remanded on appeal). *See* Amazon.com, 73 F. Supp. 2d 1228 *vacated and remanded by* 239 F.3d 1343.

[69] For a discussion of the historical and international development of legal hybrids *see* J.H. Reichman, *Legal Hybrids between the Patent and Copyright Paradigms*, 94 COLUM. L. REV. 2432 (1994).

[70] NATIONAL ACADEMY OF SCIENCES, THE DIGITAL DILEMMA (2000) [hereinafter DIGITAL DILEMMA].

[71] *See id.* at 239.

[72] *See id.* at 16. The study further suggested that when reform is contemplated, it be evaluated in line with a list of principles designed to reward creativity without impeding

The *Digital Dilemma*'s recommendations stem, at least in part, from a realistic recognition that *sui generis* software protection is unlikely to emerge in the near future, and that the likelihood of its peaceful emergence grows slimmer as the initial debates recede into history and as the commercial reliance on existing terms of protection continues to grow. A similar sense of realism helped to motivate a number of other proposals more conservative than that of the Manifesto. Cohen and Lemley, for example, argued that a narrow scope for software patents could reduce, if not eliminate, the problems posed by overprotected software.[73] They contend that this narrow scope can *probably* be achieved within the confines of several existing patent law doctrines, primarily the experimental use doctrine and the doctrine of equivalents.[74]

The experimental use doctrine allows competitors to make certain restricted uses of a patented invention in the name of experimentation, or the furtherance of knowledge.[75] Firms developing blocking patents, or patentable improvements on their competitors' patented inventions, frequently rely upon this doctrine.[76] Cohen and Lemley suggested that the experimental use doctrine could be read broadly enough to permit a limited right to reverse engineering—a right that would circumscribe a patent holder's ability to shut down its competitors' development efforts.[77]

The doctrine of equivalents defines the amount of dissimilarity required before a product competing with a patented invention is considered to be non-infringing.[78] The more broadly the doctrine is viewed, the greater the scope of the patent and the larger the portion of the software industry threatened by the patent. Cohen and Lemley contended that a narrow reading of the doctrine, at least in the context of software, is both appropriate and important to protect innovation.[79]

---

development. *See id.* at 236-38.

[73] *See* Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 1, 5-6 (2001).

[74] *See id.* at 7, 36-37.

[75] "Experimental use" is a well-recognized, judicially created defense to infringement in cases in which the patented product was used solely for research or experimentation without the authorization of the patent holder. *See* MERGES ET AL., *supra* note 9 at 295-97.

[76] *E.g.*, City of Elizabeth v. American Nicholson Pavement Co., 97 U.S. 126 (1878) (holding that an inventor's public use of an invention to test its qualities was not a public use within the meaning of the statute); Pharmacia, Inc. v. Frigitronics, Inc., 726 F. Supp 876, 885-86 (D. Mass. 1989) (holding that sale of an experimental product was by definition noncommercial and not subject to the section 102(b) statutory bar).

[77] *See* Cohen & Lemley, *supra* note 73, at 29-30.

[78] "The doctrine of equivalents allows a patent owner to hold as an infringement a product or process that does not correspond to the literal terms of a patent's claim but performs substantially the same function in substantially the same way to obtain the same result as the claimed subject matter." CHISUM ON PATENTS, *supra* note 18, at § 18.04.

[79] *See* Cohen & Lemley, *supra* note 73, at 52-53.

The generally conservative tenor of such proposals is reflected in the authors' contention that necessary reform can be achieved within the scope of existing legal doctrine.[80] Cohen and Lemley admit that more radical steps may be necessary; if the courts decide that their proposed doctrinal readings are *inconsistent* with existing law, the authors advocate legislatively granted exemptions.[81] For a variety of reasons, the legislative approach may be preferable; as long as the IP laws put software in the same one-size-fits-all basket with other industries, doctrines tailored to one end of the basket may have unforeseen consequences at the other end.[82]

The need for *intelligent* reform may thus be summarized as follows: The conferral of an IP right enables various types of anticompetitive behavior.[83] While it is possible to simply grant the rights, assume that they will be wielded responsibly, and allow antitrust law to clean up inappropriate uses, such an approach is exceedingly dangerous. Society would be better served by an IP system that conferred rights that provide appropriate motivation and few opportunities for abuse. The potential hazards of the *wrong* reform are equally obvious. They could destroy the current strengths of a thriving software industry while offering little of value in return. These dual needs may be described as calling for "cautious, but potentially radical" reform—in line with the recommendations of the *Digital Dilemma*.[84]

This section has outlined a few significant proposals that have been forwarded to meet the challenge of intelligent reform. One of this article's foci is fostering an understanding of these proposals' potential to both help and harm the industry. Any such analysis must consider the full range of each proposal's costs and benefits. The analysis must recognize that the costs inherent in an IP regime are not restricted to the balance between risks and rewards that they confer on rights holders, or even on the societal costs and benefits of progress. They also include both *transaction* costs (*i.e.*, the ongoing costs of implementing and running a policy regime) and *transition* costs (*i.e.*, the potentially large one-time costs inherent in moving from one system of rights to another).[85] All of these costs can be significant, and all are

---

[80] *See id.* at 7.

[81] *See id.* at 37.

[82] It is hard to find statutory authority for the courts to interpret patent law differently in diverse industries. Narrow readings of the doctrines in the software industry, where the readings appear to be appropriate, may have significant negative consequences in other industries. This sort of "bleed through" is a potentially hazardous side effect of *all* conservative reforms.

[83] For example, IP rights granted without sufficient attention to the societal good that they claim to be serving create opportunities for potentially dangerous anticompetitive acts. These acts can have serious, detrimental effects both on consumers (who may be restricted to low quality, high-priced goods) and on competing innovators (who may discover significant barriers to innovation).

[84] *See* DIGITAL DILEMMA, *supra* note 70, at 12-16.

[85] One other potential pitfall of a transition is that industry lobbyists may view periods of

relevant to the ultimate attractiveness of a proposed reform. They are distinct, however, from the merits of a proposed regime change. The next few sections develop a framework within which such an analysis may be conducted.

III. A STATEMENT OF FIRST PRINCIPLES

The IP clause of the U.S. Constitution empowers Congress to "promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."[86] This charge makes no reference to patents, copyrights, or any other specific form of protection. It simply states a goal (*i.e.*, the promotion of art and science) and a mechanism (*i.e.*, the reservation of exclusive rights), and leaves the details to Congress. It does, however, recognize that authors and inventors constitute distinct classes, and hints that the rights reserved to these two classes of innovators need not be identical.

A return to these first principles must similarly begin with the goal of promoting innovation and with the mechanism of reserving exclusive rights. In addition to recognizing the distinction between authors and inventors, a contemporary analysis might posit the existence of multiple categories of innovators, each of whom could respond to different sorts of incentives. The goal of any first principles approach is to understand how these categories can be characterized, how the contours of the categories affect incentives, and how incentives can be structured to promote innovation and development in each category.

This first principles approach contrasts with those taken by previous IP reform proposals. The more conservative of the proposals, or those that attempt to resolve the protection of software using only existing doctrines of patent or copyright law (or minor modifications thereof), take pragmatism as their starting point.[87] Others (notably the Manifesto) begin with a consideration of software, and see computer programs as a new class of innovations that were created by neither authors nor inventors as those terms are generally understood.[88] They thus tend to propose *sui generis* forms of protection for software without questioning the broader applicability of either the patent or the copyright systems.

The implications of a first principles approach are potentially even more radical. This article's key premise is that while the software industry is among

---

legislative deliberation as opportunities to divert IP regimes away from the public good and towards the parochial concerns of their clients. The industry-specific approach advocated in this article may lead to an increase in such behavior, as rights targeted to a specific industry are likely to receive less careful scrutiny than broadly applicable rights—thereby increasing the opportunities for domination by the affected industry. Such costs are but one example of transition costs that need to be considered during any debate over reform.

[86] U.S. CONST. art. I, § 8, cl. 8.

[87] *See* Cohen & Lemley, *supra* note 73, at 7.

[88] *See* Manifesto, *supra* note 11, at 2332, 2376.

the first victims of the one-size-fits-all[89] mentality that has long pervaded thinking about IP rights, it is unlikely to be the only one. While much of the article's analysis focuses on software, many other parts are broadly applicable to other industries whose peculiarities challenge the long-standing partitioning of IP rights.

New formats of IP rights should emerge over the next few decades to address the needs of industries in which traditional IP rights are likely to be either overprotective and prone to anticompetitive abuse or underprotective and likely to result in underinvestment. The first principles approach thus leads to the following key question: *What set of exclusive rights would motivate the optimal level of innovation among the members of a given industry?*

IV. INCENTIVES, INVESTMENTS, AND INNOVATION

The key question, as stated above, remains rather abstract. This section will develop the machinery necessary to evaluate both protective regimes that offer exclusive rights and specific industries—as well as to understand the meaning of "optimality."

A. *Tradeoffs Inherent in IP Rights*

"Intellectual Property" is an artificial and counterintuitive construct. An "intellectual good" is nothing more than an idea. Ideas are non-rivalrous and non-excludable—textbook characteristics of public goods.[90] Stated in somewhat simpler terms, one person's use of an idea neither precludes anyone else's simultaneous use nor damages the idea in any way that detracts from anyone else's later use. Thus, society as a whole would be best served by the rapid and free dissemination of all newly conceived ideas, so that maximum productive use of the idea could be made at the earliest possible moment. The creation of property rights in ideas impedes their dissemination at an immediate societal cost. It also imposes a more tangible cost on society. The holder of a property right in an idea may charge for access to that idea.[91] By conferring IP rights, society—in the guise of consumers—agrees to pay monopoly rents to the right holder for the life of the grant.[92]

There is a substantial literature on the economics of public goods.[93] Part of this literature focuses on market failures, or ways in which standard market

---

[89] While a technically correct statement would admit that a few sizes fit all, the colloquially accepted expression is close enough for purposes of this discussion.

[90] *See* Menell, *supra* note 20, at 1337 (indicating that ideas are non-excludable because one cannot exclude those who pay for the use of the idea from those who do not and are non-rivalrous because having a larger and larger number of people using them does not affect the supply of the idea).

[91] *See* MERGES ET AL., *supra* note 9, at 12-16.

[92] *See* Menell, *supra* note 20, at 1340.

[93] Public goods and market failures are standard topics in a course on microeconomic

principles do not apply to public goods.[94] Ideas suffer from several of these classic market failures. If all new ideas are spread quickly and freely, innovators will have few opportunities to capitalize on their innovations, and investors will have minimal incentive to invest in innovation. Most people looking for an investment venue will choose to put their time, effort, and/or capital into tangible property that can be resold at a personal profit rather than into ideas that will benefit society at large but whose promised personal returns are limited.[95] A societal insistence on the immediate public ownership of innovative ideas will necessarily reduce private investment in innovation and will consequently reduce the number of new ideas generated.

This inherent tension has led most contemporary developed societies to create limited types of private property rights in the realm of ideas.[96] The grant

---

theory. Most good textbooks should contain useful discussions—although they may be intertwined with more general analyses of market goods and the circumstances under which market principles work. *E.g.*, DAVID M. KREPS, A COURSE IN MICROECONOMIC THEORY (1990). The issues take on a particular relevance when the focus turns to the specific problems of valuing and/or managing public goods. *See, e.g.*, ALFRED E. KAHN, THE ECONOMICS OF REGULATION (1988); ROBERT C. MITCHELL & RICHARD T. CARSON, USING SURVEYS TO VALUE PUBLIC GOODS: THE CONTINGENT VALUATION METHOD (1989). For a broad overview of the relationship among public goods, market failures, and various legal doctrines *see* A. MITCHELL POLINSKY, AN INTRODUCTION TO LAW AND ECONOMICS (1989). For a discussion of their relationship to IP law *see* RICHARD A. POSNER, ECONOMIC ANALYSIS OF THE LAW 43-50 (1998). These topics have recently emerged as important issues at the forefront of antitrust analysis, as part of the "Post-Chicago" school. *See* Carl Shapiro, *Aftermarkets and Consumer Welfare: Making Sense of Kodak* 63 ANTITRUST L.J. 483, 484-85 (1995).

[94] *See* MITCHELL & CARSON, *supra* note 93, at 1-2; POLINSKY, *supra* note 93, at 135-38.

[95] This predictable preference would also represent a cost to society, albeit one that is harder to measure.

[96] The decision to grant such rights is fairly modern (in historical terms), is not necessarily obvious, and remains a matter of some controversy at the international level. Even today, many commentators continue to advocate replacing the patent system with a system of fixed rewards. For an introductory description of 19th century critics of the patent system *see* STEVEN SHAVELL & TANGUY VAN YPERSELE**,** REWARDS VERSUS INTELLECTUAL PROPERTY RIGHTS 1-4 (Nat'l Bureau of Econ. Research, Working Paper No. 6956 (1999)). Shavell and van Ypersele also developed a model for comparing the societal benefits of a reward system (*i.e.*, where the government provides innovators with a fixed reward) and our existing patent regime. *See id.* at 8-13. They showed that given a number of assumptions about the availability of information, a reward system would be preferable to patent rights (although a system that offered innovators a choice of property rights or an award would be preferable to either pure system). *See id.* at 17-18. *But see* Brett Frischmann, *Innovation and Institutions: Rethinking the Economics of U.S. Science and Technology Policy*, 24 Vt. L. Rev. 347, 349-50 (2000) (pointing out that under current U.S. policy many innovators receive *both* rewards (*i.e.*, as research grants) and property rights (*i.e.*, as patents or copyrights)). Frischmann viewed this duality as inherently overprotective, and called for a significant rethinking and reformation of science and technology policy. *See id.* at 351-52. The propriety of property rights in ideas is of more than academic significance, however.

of IP rights thus represents a societal decision to privatize a public good.[97] Under any type of IP regime one individual owner is given some right to dictate how, by whom, and under what terms an idea is used. That owner receives a distinct benefit, while society at large absorbs the (monetary and non-monetary) costs of having the idea removed from the public realm.[98] Society is only willing to assume that cost because it believes that it receives a concomitant benefit of greater value—specifically, increased innovation.[99] IP rights provide potential investors in innovation with a mechanism for profiting from their investments. IP rights thus represent a societal attempt to harness the profit motive in order to motivate innovation.

Any assessment of the effectiveness of an IP regime must consider two perspectives and the tradeoffs relevant to them. From the *societal* perspective, benefits are accrued when an innovator develops a new idea. Costs are incurred when rights are granted to that innovator after the idea has been developed, and rents are extracted from the consumers who comprise society. From the perspective of the *individual innovator*, costs are incurred in the development of an idea—which may or may not work. Benefits are accrued by taking advantage of the rights granted to a successful innovation (*i.e.*, by charging the allowable rents). These tradeoffs provide a conceptual framework within which a *societally optimal* IP regime may be considered and developed.

B. *Societal Optimality*

The notion of a societally optimal IP regime may appear rather abstract. The basic definition of optimality follows from some fairly standard economic definitions. The societal *value* of an IP regime is the net difference between the costs that society bears to develop and to run the regime and the benefits that society accrues by establishing the regime.[100] The societally *optimal* regime is the one that maximizes societal value. A series of illustrations may be helpful to illustrate these definitions.

Consider Zero-IP (ZIP), a society that does not recognize any private rights in intellectual goods. In ZIP, some people will innovate for the sake of innovation (*i.e.*, they may simply enjoy the intellectual stimulation inherent in innovation), while others will innovate to address their own personal needs. Some market factors, such as the first mover advantage (*i.e.*, the observation that the first firm in a market is often able to establish a market niche that later

---

The rift between the developed and developing worlds' views of IP rights rose to the fore during negotiations over the GATT/TRIPS. *See* JOHN H. JACKSON ET AL., LEGAL PROBLEMS OF INTERNATIONAL ECONOMIC RELATIONS 848-850 (3d ed. 1995); Dennis S. Karjala, *Policy Considerations: Theoretical Foundations for the Protection of Computer Programs in Developing Countries*, 13 UCLA PAC. BASIN L.J. 179, 189-90 (1994).

[97] *See* MERGES ET AL., *supra* note 9, at 16-17.

[98] *See id.*

[99] *See id.*

[100] *See id.* at 15.

competitors are unable to shake) and/or the rewards available for teaching and training, provide further incentives to innovate in ZIP.

These inherent motives for innovation ensure that even though ZIP confers no property rights on its innovators, it will not be a society devoid of innovation. The innovations generated in ZIP define a *base level* of innovation. Because any society could gain access to these base level intellectual goods without awarding any private rights, any rights granted to private innovators in these goods constitute pure costs that were not strictly necessary. ZIP is thus a conservative, risk averse society that refuses to invest in innovation, but that is willing to free ride on the investments of private innovators. With no societal investments or likelihood of positive returns, ZIP-like regimes confer limited positive (or at least non-negative) values on societies that adopt them. These values, however, are unlikely to be very large. Societies willing to incur some risks by absorbing some costs should be able to generate greater positive returns.

The members of Weak-IP (WIP) societies recognize the potential value of taking some risks and introduce a weak set of IP rights. WIP's decision represents an immediate absorption of some costs; base level innovators are rewarded for tasks that they would have undertaken even in the absence of individual property rights. WIP is thus immediately worse off than ZIP, unless the rights motivate the diversion of private resources towards further innovation leading to useful ideas above the base-level innovations. If that diversion occurs, WIP will accrue restricted benefits (*i.e.*, uses that do not conflict with the private rights granted and the ability to buy the other rights back from the right holder) in two classes of innovation: (i) the base-level innovations of ZIP; and (ii) the second-level innovations that exist in WIP but not ZIP. In exchange, WIP must cede the difference in utility between unrestricted and restricted use of the base-level innovations. If the value of restricted use of the second-level innovations exceeds the reduced value of the base-level innovations, WIP will achieve a higher return in net societal value than did ZIP. Otherwise, ZIP is better off, and WIP's decision was a mistake.

If WIP is better off than ZIP, then perhaps Strong-IP (SIP) might decide to incur even further costs by adopting a stronger set of IP rights. Once again, the incremental increase in the *private* value (*i.e.*, the value that SIP's rights confer on private parties that WIP's did not) will motivate at least some additional potential innovators to develop third-level innovations. SIP will thus accrue a net societal benefit equal to the amount that the newly-restricted rights on the three levels of innovation exceeds the less-restricted rights on the first two levels of innovation.

This pattern of costs and benefits will continue as the conceptually incremental process of strengthening private IP rights progresses, but only to a point. Early in the strengthening process, increased rights can be expected to spur additional innovation. Eventually, however, the rights may become so expansive that they block innovation. New entrants may become discouraged when virtually anything that they discover infringes a right that has already been granted. Thus, societies that grant increasingly stronger rights may gain

increasingly restricted use of a growing pool of innovations—or they may deter future innovation. As each strengthening proposal is considered, society must ask itself whether the pool is likely to grow or to contract. If it is expected to grow, society must ask whether the increased restrictions across the larger pool are likely to result in a net gain or a net loss. Proposals that promise a net gain should be adopted; those that promise either a net loss or a smaller pool of innovations should be rejected. When no available proposals promise to yield a net benefit, the regime in place is societally optimal.[101]

C. *Parameters of Protective Strength*

Optimal protection is harder to recognize than it is to define. The ZIP-WIP-SIP metaphor essentially glossed over the meaning of "weak" and "strong" IP rights. A true comparison of competing IP regimes requires mechanisms for measuring strength, costs, and benefits. Three protective dimensions are well suited for this task: *breadth*, *depth*, and *length*.

The *breadth* of protection refers to the similarity between a protected product and a competing product that is required before the rights holder can claim that her rights have been infringed. In the narrowest possible regime, identical products infringe these rights, but products embodying even *de minimis* differences do not. In the broadest possible regime, even a *de minimis* similarity constitutes infringement. All reasonable IP regimes fall somewhere between these extremes. The broader the regime, the greater the value conferred to the private innovator and the greater the cost borne by society.

The *depth* of protection refers to the uses that the holder of an IP right may restrict. Deep regimes allow owners to restrict many activities, including potentially severe restrictions on resale. Shallow regimes confer the right to restrict only a few uses. Again, the deeper the regime, the greater the value promised to potential innovators and the greater the cost to society.

The *length* of protection refers to the period of time over which the holder of an IP right may restrict the public use of her innovation. Patents and copyrights, for example, are both of limited length. Under both regimes, owners are permitted to restrict the granted breadth and depth of public use of their innovation throughout the terms of the protection, and not at all after the right's expiration. Other approaches to length are also possible; a property right could confer different rights as it aged. For any fixed form of protection, longer terms are strictly more valuable to private innovators and more costly to society.

---

[101] As a technical matter, this situation describes a *local* optimum, not necessarily a *global* optimum. The Optimal-IP society that adopts this regime has no reason to absorb the further costs implicit in granting increasingly restrictive IP rights. A society that continues to strengthen its IP rights beyond that point is *overprotecting* intellectual property. A society that sees available net benefits in increasing IP rights but fails to strengthen them is *underprotecting* intellectual property. Those that adopt all proposals that promise a net positive return—and *only* proposals that promise a net positive return—are *optimally protecting* intellectual property.

The meaning of protective length is relatively straightforward, and it is easy to see how varying the length of protection can increase or decrease the overall strength of an IP right. On the other hand, characterizing breadth and depth variations may be quite complicated. Patents, for example, provide deep protection with respect to commercialization,[102] but rather shallow protection of the underlying knowledge.[103] Copyrights are at least as shallow as patents in their protection of the underlying knowledge,[104] but quite broad in the protection that they afford to concrete representations of that knowledge.[105] While the development of patent and copyright law has tinkered with the fringes of these parameters, the basic description of commercially deep patents and broad copyrights appears to stem from the nature of the innovations that they were designed to motivate. Substantial variations of overall strength are thus most likely to occur along the subtler parameters of patent breadth and copyright depth. Existing patent and copyright law doctrines can help to demonstrate such potential variations.[106]

1. Breadth

How broad is a patent? Stated another way, how much similarity is required to trigger infringement? In some sense, infringement is often difficult to prove. Many patents are drawn quite narrowly, and a successful infringement suit must prove similarity of all elements.[107] The challenges inherent to a successful suit notwithstanding, the fundamental underlying question remains: How similar is substantially similar? And more to the point, how can variations in the required level of similarity be used to craft regimes of different strength?

The doctrine of equivalents, a standard patent doctrine that embodies the variable-breadth concept, provides an illustrative answer to both questions.[108] This doctrine emerged when the courts recognized that if infringement required *literal* duplication of every aspect of a patented invention, copiers would discover ways to introduce insignificant changes into their products to circumvent patent protection. Were this type of copying not viewed as infringement, the patent right would be essentially useless. The doctrine of

---

[102] The patent holder retains the right to impose fairly severe restrictions on distribution, sales, repair, etc. *See* discussion in *supra* note 9.

[103] Any practitioner of the relevant discipline is supposed to be able to understand the published patent and incorporate its contents into her own work. *See* 35 U.S.C. § 112 (2000).

[104] This statement applies to traditional, non-functional copyrights, such as text and artwork. The difficulties introduced by functional copyrights, such as those awarded on software, are discussed at length in § VI.B.1.

[105] Even small similarities between representations can constitute infringement.

[106] *See infra* §§ IV.C.1 and IV.C.2.

[107] *See* UNDERSTANDING INTELLECTUAL PROPERTY LAW, *supra* note 7, at § 2F[2][a].

[108] *See supra* note 78 (defining the doctrine of equivalents).

equivalents permits a patent holder to proceed against the producer of a device that "performs substantially the same function in substantially the same way to obtain the same result."[109] Terms like "substantially the same," however, are subject to interpretation. The liberality with which they are interpreted defines, in part, the breadth of the patent protection.

The debate between the majority and the dissent in *Graver Tank*, a classic doctrine of equivalents case, is illustrative.[110] The respondent, Linde, held a patent for an electric welding process that used a patented composition containing two alkaline earth metal silicates: calcium and magnesium. Graver developed a process that was like Linde's in all respects but one; its welding composition contained calcium and manganese silicates.[111] Manganese is not an alkaline earth metal.[112] The majority applied the doctrine of equivalents to accept the trial court's finding of infringement.[113] The dissent, however, viewed the Linde process's reliance on alkaline earth metals as central to the patent protection, and would have found for the defendant.[114] This debate can be recast in terms of the breadth of patent protection. The majority took a more liberal view of equivalence and thus a broader view of the patent rights than did the dissent, showing that patents are broad in part because the doctrine of equivalents insures that a sizable number of differences between the original item and competing ones are required for a finding of non-infringement.

The debate implicit in *Graver Tank* also has differing implications to the realms of law and of public policy. From a legal perspective, the question must be where the line between infringement and non-infringement *is*. From a policy perspective, the question is where the lines *could* be or *should* be. The debate over the appropriate definition of "equivalence" is representative of this struggle. Many other patent law doctrines address comparable questions about the placement of the line between infringement and non-infringement.[115] Each possible answer has different policy implications and may warrant consideration within a first principles framework. Each possible answer also defines a different breadth of protection.

---

[109] Sanitary Refrigerator Co. v. Winters, 280 U.S. 30, 42 (1929).

[110] *Compare* Graver Tank & Mfg. Co. v. Linde Air Products Co., 339 U.S. 605, 612 (1950), *reh'g denied,* 340 U.S. 845 (1950) *with id*. at 616 (Black, J., dissenting).

[111] *See id*. at 610.

[112] *See id*. at 618 (Douglas, J., dissenting).

[113] *See id*. at 612.

[114] *See id*. at 616 (Black, J., dissenting).

[115] To cite but a few examples, experimental use allows the otherwise unauthorized use of patented articles for research or experimentation. *See* CHISUM ON PATENTS, *supra* note 18, at § 16.03[1]. The first-sale doctrine precludes patent holders from conditioning the use or resale of a patented product. *See id.* at § 16.03[2]. The repair and reconstruction doctrine allows the purchaser of a patented article to effect the repairs necessary for the article's continued use. *See id*. at § 16.03[3]. Alternative IP regimes could reverse any of these doctrines and rule that the behavior that they allow constitutes infringement.

2. Depth

The depth of a patent, as noted above, must be characterized differently for knowledge and for commercialization. The patent system was designed to insure that the *knowledge* embodied in the patented good is disseminated widely, while the commercial exploitation of that knowledge by anyone other than the patent holder is severely restricted.[116] This protection may be viewed as shallow for scientific purposes but deep for commercial purposes.[117]

Copyrights are even shallower than patents in their protection of knowledge,[118] but considerably broader in their protection of expression. Many similarities deemed to be copyright infringements would not be similar enough for patent infringement. This distinction recognizes that while a patent holder contributed a new *idea*, a typical copyright holder contributed an *expression* of an underlying idea that may already have been well understood.[119] Thus, society allows patent holders to restrict commerce in their idea—even if it is expressed in a different way—while copyright holders may restrict commerce in their expression—even if it is used to represent a different idea.

The right to restrict the use of a representation raises some interesting depth questions. Should the copyright holder, for example, be allowed to restrict the circulation of a textual or artistic creation? After all, the context in which a work is viewed, or even the mood or emotional state of the viewer, can have a dramatic impact on the way that the representation is perceived. Copyright law has addressed these questions by developing a number of doctrines that limit depth and illustrate the potential for depth variations that could alter overall protective strength. One such doctrine is fair use.[120]

---

[116] *See* MERGES ET AL., *supra* note 9, at 137. Patent law is based in part on the theory that while inventions are public goods, an appropriate incentive system is needed to convince private parties to bear the cost of their development. Patent law provides this incentive by allowing an inventor to "appropriate the full economic rewards of her invention." *Id.* The requisite public disclosure of all patented inventions provides the appropriate counterbalance to this private benefit. *See id.* at 23.

[117] By way of contrast, conventional trade secret law is both narrow and shallow. It is narrow because a successful reverse engineering effort renders the original developer shorn of all rights. The developer, however, is under no obligation to aid the reverse engineering. Any legal step taken to secure the secret is considered legitimate. Thus "protection" of a secret is as deep as the developer can make it using individual efforts, and not broad at all. *See* MERGES ET AL., *supra* note 9, at 22-23.

[118] Again, this statement refers only to traditional, non-functional copyrights.

[119] *See* UNDERSTANDING INTELLECTUAL PROPERTY LAW, *supra* note 7, at § 1B[1] and [3]. Patentable subject matter includes any new and useful process, machine, manufacture, or composition of matter, *see id* at § 1B[1], while a work may be copyrightable without being new or even very different from prior creations. *See id.* at § 1B[3].

[120] *See* 17 U.S.C. § 107 (2000) ("[T]he fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple

The fair use doctrine permits everyone to make certain "fair uses" of copyrighted material without seeking authorization from or offering compensation to the copyright holder.[121] The doctrine thus immediately limits the depth of protection. Case law has developed to determine where that limitation lies.[122] An individual who purchased this journal, for example, is permitted to make a single photocopy of this article for his personal use. A corporate subscriber, on the other hand, may be prohibited from circulating multiple copies of this article among its employees.[123] For-profit copy shops may be similarly restricted.[124] Regardless of the limitations set by the fair use doctrine, libraries and archives follow a different set of statutory rules that allow them to produce limited numbers of copies for specific uses without explicit authorization.[125]

These distinctions illustrate the general concept of variable depth by showing how the line between permissible and impermissible copying can be drawn in a variety of places. It is easy to imagine copyright regimes in which libraries and archives are prohibited from unauthorized copying, corporations are allowed to circulate copies to their employees, and commercial copying shops are given more (or less) leeway in their copying practices. Each of these regimes would define a different depth of protection.

Under the current regime, the commercial protection afforded to the ideas embodied in conventional (*i.e.*, non-functional) copyrights is both narrower and shallower than that given on patents. Like patents, conventional copyrights reveal their underlying knowledge for all to see; competitors can generally "reverse engineer" a copyrighted work simply by reading, viewing, or listening to it. Unlike patents, however, conventional copyrights allow competitors to use the ideas underlying the copyrighted work in their own

---

copies for classroom use), scholarship, or research, is not an infringement of the copyright.").

[121] *See* 17 U.S.C. § 107 (2000); MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT, at § 13.05 (discussing the fair use doctrine).

[122] *See* Princeton Univ. Press v. Michigan Document Servs., 99 F.3d 1381, 1383 (6th Cir. 1996) (en banc), *cert. denied*, 520 U.S. 1156 (1997) (holding that a for-profit copy shop could not make copies of copyrighted books for use in course packets sold to students); American Geophysical Union v. Texaco, Inc. 60 F.3d 913, 931 (2d Cir. 1994), *cert. denied*, 516 U.S. 1005 (1995) (ruling that a corporate subscriber to a professional journal did not have the right to photocopy articles to circulate among its employees); Basic Books, Inc. v. Kinko's Graphics Corp., 758 F. Supp. 1522, 1526 (S.D.N.Y. 1991) (finding infringement by a for-profit copy shop producing course packets from copyrighted materials).

[123] *See* American Geophysical, 60 F.3d at 931.

[124] *See Princeton Univ. Press*, 99 F.3d at 1383; *Basic Books, Inc.*, 758 F. Supp. at 1526.

[125] *See* 17 U.S.C. § 108 ("Notwithstanding the provisions of section 106, it is not an infringement of copyright for a library or archives, or any of its employees acting within the scope of their employment, to reproduce no more than one copy or phonorecord of a work, or to distribute such copy or phonorecord under the conditions specified by this section . . .").

innovations. Protection is extended only to the literal elements of the work and to a relatively small class of derivative works, *not* to the underlying idea.[126]

3. Interaction among the Parameters

Beyond the illustrations of the protective parameters drawn from patent and copyright law, it is important to recognize that breadth, depth, and length are essentially independent dimensions. A societal decision about breadth of protection, for example, does not necessarily restrict decisions about depth and length. IP regimes may thus be parameterized according to their "three-dimensional" scores.[127] The total strength (hence value) of an IP regime is a function of all three parameters. While the interplay among the parameters is complex and likely to differ across industries,[128] the notion that a regime's value may be expressed parametrically suggests that there are multiple ways to generate desired quanta of IP protection.[129] Stated another way, society can motivate the same amount of innovation in different ways. The configuration of the protection can direct investment towards some industries and away from others.

This last point can be restated somewhat less technically with the help of an illustrative tradeoff between two parameters. Consider a regime that requires very substantial similarity for infringement, but that prohibits unauthorized use even for personal, non-commercial purposes (*i.e.*, a fairly narrow, deep regime). One contemplated reform might keep the length of protection fixed, slightly broaden the range of similarities across which infringement is found, but allow personal, non-commercial use. This proposal, if adopted, would

---

[126] *See* 17 U.S.C. § 102(b) ("In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.").

[127] Conceptual parametric scoring is not intended as a strict guide to policy. Policy should not be decided on the sorts of narrow, technical grounds that might be inferred from a strict reading of this discussion. Nevertheless, a conceptual parameterization can and should direct the analysis of both an existing policy regime and any proposed reforms. Section VII *infra* provides a worked example of this guidance.

[128] In many industries, the first few years of protection are generally considered to be the most valuable. After all, consumer demand is generally greatest for new technologies. By the time that a patent reaches its latter years, future generations and/or superior technologies are likely to have overtaken the protected invention. The pharmaceutical industry provides a stark exception to this rule. Drug companies are prohibited from marketing their discoveries until they have gained clearance from the Food and Drug Administration—a process that can consume more than half of a product's protected lifetime. As a result, the value of a pharmaceutical patent is likely to be greatest in its final years. In still other industries, the first few years of the protection may be devoted to generating consumer acceptance and/or motivating the development of complementary products. In those cases, the middle years are likely to be the most lucrative.

[129] Although it is not necessarily true that any arbitrary combination of parameters is achievable.

have two obvious consequences. Some potential innovators who chose not to invest their time, effort, and resources under the old, narrow regime might now choose to do so in the new regime because of the increased breadth. At the same time, some old-regime innovators might choose to refocus their efforts away from innovation because of the new, shallower protection being offered. If the investment attracted equals the investment deterred, these two regimes confer equivalent benefits on society.[130]

In addition, regimes that confer equivalent societal benefits need not be of equal cost. Societal benefits, you will recall, are measured in terms of innovation attracted, while societal costs are measured in terms of restrictions accepted.[131] If the societal cost of foregoing personal, non-commercial use exceeds the cost of broadening the range of infringing products, the proposed change in the regime promises to maintain societal benefits while cutting societal costs, thereby increasing the net value to society.

This illustration highlights the challenge of recognizing optimally protective regimes. It also suggests another way to think about the degree of radicalism inherent in a reform proposal. Under an incremental (or conservative) proposal, a single parameter is adjusted. Costs and values can both be forecast by interviewing groups of affected stakeholders. If the proposed shift promises a net benefit, it should be adopted; otherwise it should be rejected. Under a radical proposal, two or more parameters are adjusted simultaneously, thereby presenting both a much tougher analytic challenge and a greater likelihood of unintended consequences.[132]

This characterization of incremental and radical reform is hardly unique to IP regimes. As a general rule, incremental reforms are easier to propose, to assess, and even to retract when necessary than are radical reforms. Radicalism is often required, however, when an existing regime or policy appears to be both inadequate and beyond fixing. A regime with these properties may be viewed as stuck at an unacceptable local optimum. In the context of IP protection, a locally optimal regime is one in which no incremental change to any single parameter will either reduce societal costs or increase societal value. An unacceptable local optimum is one in which the net benefits conferred to society by the regime are insufficient to justify the societal costs.[133] When a policy regime gets stuck at an unacceptable local optimum, radical reform is necessary, and only major systemic changes stand a chance of reaching an acceptable solution.[134]

---

[130] These benefits are unlikely to be identical because the specific innovations that they motivate are likely to differ.

[131] *See* discussion in *supra* § IV.A.

[132] While this characterization of radicalism differs from the one presented in § II, it is really just a variation on the theme. Conservative proposals that operate within existing doctrines tend to vary one doctrine—hence one parameter—at a time. Radical *sui generis* or first principle proposals are likely to vary multiple parameters simultaneously.

[133] In a catastrophic case, the costs may even exceed the benefits.

[134] The distinction between incremental and radical reforms may be described in less

A further factor complicating potential reforms to an IP regime emerges from the law of diminishing returns, an empirical economic observation that the addition of fixed amounts of an input tends to have a progressively decreasing impact as the amount of that input present increases. A concrete example in the IP context is that the value of an added year of protection is likely to be greatest for rights protected for a short length of time, and least for those already protected for a long period of time. The benefits to society in strengthening the private rights provided to potential innovators along any parameter are thus likely to be inversely proportional to the magnitude of the restrictions that society has already allowed along that parameter. Under the law of diminishing returns, the greater the private rights already granted by an IP regime, the greater the reform necessary to effect a desired change in behavior. This need is present in both incremental and radical reforms. Incremental reforms must add progressively larger quanta along the adjusted dimension, while radical reforms must become progressively more radical.

D. *Private Value*

The discussion thus far has focused on the costs and benefits to society. The societal perspective on IP rights, however, is only part of the equation. The rest lies in the private sector. Private sector innovators are motivated by a variety of factors. Some innovate for the simple love of their art and consider monetary rewards to be secondary. These innovators are driven by a desire to expand truth, beauty, and knowledge and are often oblivious to the applicability and potential profitability of their work. While IP rights are not necessary to motivate such innovators, a poorly constructed regime might deter

---

technical terms. Consider some area regulated by the federal government that has been subject to radical change at least once over the past twenty years (*e.g.*, taxes, welfare, communications). Congress began its regulatory involvement in these arenas by passing a bill into law. Over the years, legislators considered a variety of "conservative reforms" to the existing statute—largely localized incremental changes designed to achieve some desired goal. Those that passed were, by definition, those that a majority of Congress viewed as an improvement over the situation *ex ante*. In other words, every change was seen as a local improvement. In many cases, a broad consensus recognized that these sequential local improvements had led to a Byzantine and unworkable regulatory code. No small patch seemed to offer a substantial improvement. Reform to these regulations is only possible when the various stakeholders agree to a major overhaul of the entire regulatory system. Such "radical reforms" corresponded to a simultaneous change of multiple parameters. (For three examples, consider the reforms of the income tax code in 1986, to the welfare system in 1996, and to the telecommunications industry in both 1984 and 1996). The immediate effect of these changes can be chaotic. Conservative patches applied to the new system are often required to ameliorate these problems. Taken together, then, conservative and radical reforms are symbiotic. A radical step is often needed to change the fundamental structure of a flawed system. It is naïve to believe, however, that the outcome of a radical restructuring will lead to an immediate improvement. Instead, one goal of the radical phase should be to establish mechanisms whereby conservative fine-tuning can develop a superior system.

them. These innovators can generally be motivated by fixed-price grants or contracts (either from the government or from private patrons) that allow them to ensure that their bills are paid while they innovate, rather than by the more open-ended concerns of an IP system. Such innovators-for-the-love-of-it are therefore not the focus of the current analysis.

A much larger component of the private sector is composed of potential innovators and potential investors in innovation whose primary objective is to see a return on their investment.[135] If society would like to see these resources directed towards innovation, the investors must be able to project a suitable return. The resources will thus only be devoted to innovative pursuits if each potential investor believes *both* that the expected rewards of investing in innovation will exceed the expected costs of that investment *and* that the expected net benefits of investing in innovation will exceed the expected net benefits of other available investment opportunities. These requirements are non-trivial. After all, innovation is a necessarily risky venture, and society remains unwilling to grant *unrestricted* rights to an innovator.[136] Furthermore, society would like to achieve these goals within the context of an optimal IP regime, not an overprotective one.

In order to understand how a viable IP regime can be constructed, it is important to focus on the sorts of decisions that will be encountered by individual investors. As a somewhat oversimplified model of these decisions, consider the issues facing an individual (or possibly a corporation) searching for an attractive investment of private resources. On the one hand, the investor could pursue a safe investment, such as a Treasury Bill, and receive a "risk free" return. On the other hand, the individual could invest in research and development (R&D), and receive a return if and only if the investment led to a successful innovation, *and* the innovation could be turned into a commercially successful venture. If the output of this process is tangible property or can be protected as a trade secret, the venture may proceed. If, on the other hand, the venture is a functional manifestation of an idea that cannot be kept secret, some form of IP protection must be granted before a return can materialize. In such instances, a return will accrue only if the innovator is able to secure the IP protection *and* no competitor develops that innovation (and secures IP protection) first. The investor must consider these uncertain contingencies in probabilistic terms. Their interdependence suggests that the investor's likelihood of seeing *any* return on the R&D investments is close to the product of the four probabilities.[137]

---

[135] Note that the actual innovators invest their time and effort, while their backers invest dollars. As a result, all participants in this system may be considered investors in innovation.

[136] While it is possible to argue that society *never* grants completely unrestricted property rights, it should be clear that there are more restrictions on activities allowed for intellectual property than for tangible property.

[137] The investor will see a return only if *all four* statements are true: (i) The inquiry must lead to an innovation; (ii) The innovation must lead to a product; (iii) The innovator must

The probability of seeing a return is only a small part of the story. The true measure of an investment's attractiveness is the expected magnitude of that return. Once again, the expected return on the investment is a complex function of the breadth, depth, and length of the rights that society is willing to grant. The first two of these dimensions, breadth and depth, configure the realm of commercial opportunities that the investor can consider. The commercial venture suggests a market potential and a projected price, which, in turn, allow the innovator to forecast the potential value of the innovation, if successful.

Breadth and depth also interact in a somewhat subtler way in the determination of commercial viability. In narrowly protective regimes, a competing innovator whose work lags behind that of the leader may be able to divert a large part of her investment into a similar, but non-infringing innovation. The broader the protection becomes, the harder it becomes to divert successfully without infringing. Thus, broad protection both increases the rewards to the ultimate winner and decreases the probability that each individual competing innovator will receive anything. Shallow protection provides a similar opportunity to divert partial results. Deepening the protection makes diversion correspondingly more difficult.

Determining the appropriate amount of diversion to allow is a challenging task. Diversion is essentially a distributional mechanism that reduces individual risk by shifting some of the rewards from the successful innovator to an industry at large (*i.e.*, both the successful innovator and its actual and potential competitors). Like most distribution mechanisms, diversion can have both positive and negative effects.

An IP regime that allows no diversion (*i.e.*, a broad, deep, long regime) grants 100% of all possible returns to the first successful innovator, and nothing to second- or third-place finishers. This sort of rule creates an extremely high-risk investment regime, in many ways comparable to a high-stakes lottery. Investors in innovation are invited to purchase expensive R&D "tickets," one of which at most will pay off, possibly at an extraordinary rate of

---

secure IP protection; and (iv) No competing innovator can get there first. The conditions are probabilistically dependent on each other. The technically correct specification of a probabilistic model would incorporate a set of *conditional probabilities* and then combine them to generate a *discrete global joint probability distribution*. The probability assigned to the conjunction of the four conditions within this distribution represents the probability that the investor will receive a non-zero return. Truly motivated potential investors might consider building formal statistical models of the interrelationships among these factors. The technical details of model specification and manipulation are beyond the scope of this article. Interested readers might wish to review Izhar Matzkevich & Bruce Abramson, *Decision Analytic Networks in Artificial Intelligence*, 41 MGMT. SCI. 1 (1995) and/or R.T. CLEMEN, MAKING HARD DECISIONS chs. 7-8 (2d ed. 1996). The key point to note is that each of these four conditions is uncertain, and that the promise of any return as viewed at the time of the initial investment can be rather slim. In order to motivate the investment, then, the promised return must be substantial.

return. All other investors will lose their entire stake. While this approach may motivate extreme dedication to innovative R&D, it will also deter even moderately risk-averse investors. The expected payoff may be high, but then so is the risk of a total loss.[138] Thus, some amount of diversion may be desirable to reduce the risk inherent in R&D investments.

At the same time, diversion inevitably promotes free riding. Some potential investors may feel that they can maximize their returns by investing the bulk of their resources in conservative ventures unrelated to innovation and reserve only the relatively modest amounts necessary to monitor the progress of their innovative competitors. These conservative investors can then use their competitors' innovations to develop their own products, thereby attempting to profit from investments that they themselves did not make—in other words, to free ride on their competitors' risky investments. Regimes that are too narrow and shallow can thus restrict the rights granted to the successful innovator, increase the profit opportunities available to free riders, and, consequently, reduce the attractiveness of investment in innovation. Appropriate diversion must be implemented to promote a principled mechanism that differentiates between legitimate investors in innovation, who society may wish to reward for their nearly successful efforts, and free riders, whose behavior society has no particular reason to reward.[139]

Patent law provides an example of a mechanism that attempts to achieve this goal: the *blocking patent.*[140] Suppose that firm X patents and manufactures

---

[138] Private investors are not the only ones who would lose in this extreme winner-takes-all world. While only one of the investments could possibly be the first to pass the finish line, it is likely that many of them generated useful innovations. If these research efforts are terminated and/or buried, society as a whole loses. This type of wasteful "rent seeking" behavior has a long history of concern to legal thinkers—dating back at least as far as some common law doctrines governing the rights of treasure hunters. *See* POSNER, *supra* note 93, at 41.

[139] Free ridership describes a full spectrum of behavior—with a correspondingly broad range of societal impacts. Some potential innovators who choose to sit on the sidelines during the invention stage, for example, may be able to introduce productive resources during the manufacturing stage. If they were allowed to use these resources to increase supply of the new innovation—and thus to lower price—society might be well served by allowing them to do so. The prohibition against allowing these free riders into the market is thus a societal cost that belongs in the equation evaluating the overall value of the IP regime. Extreme free ridership, however, adds no meaningful efficiencies—it simply allows consumers to choose between otherwise identical goods produced by the innovator or by a new entrant. This new entrant could be blocked without imposing any costs on society. In the extreme, then, society has no incentive to reward free riders. Beyond that, the general argument against free ridership is deterrence; if free riders are rewarded, all potential innovators are motivated to free ride. This sort of perverse incentive can have devastating societal consequences.

[140] *See* Prima Tek v. A-Roo Co., 222 F.3d 1372, 1379 n.2 (Fed. Cir. 2000) ("A blocking patent is an earlier patent that must be licensed in order to practice a later patent. This often occurs, for instance, between a pioneer patent and an improvement patent."); *see also*

widgets. While X's first-generation widgets may be fine products, there are probably ways in which they could be improved or combined with other existing products. Any firm, X or its competitors, that discovers a suitably novel improvement to the widget, may receive a patent on *that* innovation in exchange for making its R&D public. Suppose that firm Y receives a patent on an improved widget. The improvement is essentially blocked from coming to market. Y cannot market them without infringing X's patent. At the same time, X cannot market these second-generation widgets without infringing Y's patent. While it is certainly possible that X and Y will refuse to collaborate, and that the patent system will thus have deprived society of second-generation widgets, it is in both firms' rational self-interest to reach an agreement that enables the marketing of second-generation widgets. Whether the X-Y collaboration occurs or not, free rider Z, who made no investment in widget research, may not sell either first- or second-generation widgets without infringing at least one of the patents.

The blocking patent system enables both innovators to reap rewards, confers the benefits of all widget research on society and reduces free-riders to peripheral roles in the market until the expiration of the rights granted to X and Y.[141] In addition, society gains by creating competition in the widget industry.[142] In a winner-takes-all regime, X may start out as the most innovative firm in the widget industry, but once X has been awarded full, broad rights, X will have little incentive to continue innovating. Left to its own devices, X may never develop the second-generation widget. In the long run, X is likely to decide that the extraction of monopoly profits on first-generation widgets provides a steady, low risk, high return income and that further investment in R&D cannot be justified. Society, again in the personae of consumers, thus suffers a major loss when a single firm can enjoy a broad, deep, monopoly until the expiration of its IP rights.

The expiration of IP rights, in turn, is nothing more than the use of the third dimension of IP protection—namely length—to allow (or to prohibit) diversion. Even very broad, deep rights are unlikely to lead to an entrenched monopoly if they are of relatively short duration. In particular, if the rights are crafted to last only about as long as a generation of widget technology, firms who lost the competition for one generation could easily re-enter the fray for the next one. At the same time, free-riding would be deterred because free-riders would be frozen out of each successive generation with the granting of the new set of rights.[143]

MERGES ET AL., *supra* note 9, at 284-87 for a general description of blocking patents.

[141] *See id.* at 1169-71.

[142] *See id.*

[143] Neither blocking patents nor IP rights with a short shelf life can fully restore the benefits of a competitive market. In both settings, firms compete to become the next-generation monopolist. Consumers continue to pay prices that are considerably higher than they would be in a competitive market. These monopoly prices are part of the price that society has decided to pay in return for promoting innovation; in the absence of the ability to

In addition to its role as a diversion mechanism, length also plays a second, perhaps subtler, role in determining the value of the rights being conferred. IP length is uniquely important because of the time value of money.[144] Investment in innovation tends to be a time consuming process. Present resources must be invested in order to generate expected future returns. No returns are possible until the innovation has led to a commercially viable product. In some cases, the returns may have to be deferred until *both* the administrative process of receiving the applicable IP protection *and* the development of a commercial entity have been completed. Returns will then unfold over time, accruing at different rates in different industries. In some industries, for instance, demand might be explosive. Many consumers will purchase the innovative good as soon as it hits the market. In others, demand may grow incrementally. Profits will start low and build as time progresses. In either case, the forecast profits must be discounted back to their net present value in order to compare them with the amount invested.[145]

---

charge these prices, private investment would not have developed the widgets at all. The competition to become the next generation monopolist does yield an important societal benefit: it enhances the quality of the goods being sold. This tradeoff illustrates the societal considerations described in text. The IP rights allow consumers to purchase expensive, high quality goods. They also motivate commercial enterprises that further scientific progress. In the absence of these rights, the goods would be of lower quality, and the scientific progress that they embody would be lost. They would also be priced differently—possibly more expensive, possibly cheaper. If the quality improvements and scientific benefits exceed the price differential, society has benefited from the rights. Otherwise the rights are hurting society—and the policy underlying them needs to be rethought.

[144] Ignoring the possibility of a deflationary environment, there is more value in receiving $100 today than in receiving $100 at some point in the future since the $100 one receives now may be invested for profit in the interim.

[145] At the risk of oversimplification, a discount rate is the inverse of an interest rate. Thus, for example, suppose that $1 could be invested today to yield a 10% interest rate one year from now. Then in many ways, possession of $1 today and possession of $1.10 one year from today are essentially equivalent states. They simply describe different temporal points within the same transaction. In financial terms, $1 is the *present value* of the investment; $1.10 is its *future value*. Discounting is simply the financial algorithm for converting future values back into their (nominally smaller) present values. Financial theory contains several procedures for calculating appropriate discount rates under a variety of different scenarios—including those used in litigation. *See, e.g.*, James M. Patell et al., *Accumulating Damages in Litigation: The Roles of Uncertainty and Interest Rates*, 11 J. LEG. STUDIES 341 (1982); Franklin M. Fisher and R. Craig Romaine, *Janis Joplin's Yearbook and the Theory of Damages*, 5 J. OF ACCOUNTING, AUDITING, AND FINANCE 145 (1990). For present purposes, it is important to understand only that future returns—and in particular future expected returns—must be discounted back to the present before they may be compared fairly to the investment required to generate them. It is also important to recognize that the more temporally distant the returns, the larger they must be to justify a fixed current investment. Thus, unless the out years of the protected innovation are expected to generate explosive revenues, they are unlikely to add much in the way of *ex ante* incentives. Industries that do not typically anticipate such revenue patterns are thus

Taken together, then, the expected net present value of a current investment in R&D is the discounted value of the expected revenue stream that it is forecast to generate. Any R&D investment worth considering must promise returns whose expected net present values exceed the investment required. R&D investments likely to be pursued must also promise rates of return that exceed those of other, less risky investments, such as investment in an existing business or in a government bond. IP regimes designed to motivate innovation must consider all of these factors in assessing the attractiveness of the rights being offered to potential investors in innovation.

V. THE ANALYTIC FRAMEWORK

The preceding section laid the groundwork for investigating the extent to which an IP regime will promote innovation in a given industry. This section will recast that discussion in four analytic stages.

A. *The Four Analytic Stages*

The analytic framework can be conceptualized in four (potentially iterative) stages:

- Stage 1: Characterize the Industry. The aspects of the industry likely to be relevant are those that play against the protective parameters of length, breadth, and depth. Up front capital costs, the inherent riskiness of research efforts, and the number of innovations needed to field a product are basic characteristics of the industry. Expected time from investment to return is the key factor relevant to length. Existing industry participants, likely entrants, and barriers to entry are relevant to breadth. Ease of copying, ease of use, needs for training, complementary equipment, etc. are relevant to depth. Again, some of these issues may require detailed discussion and analysis, but general rules should be observable in any mature industry, and trends and expectations should appear in the literature for young and/or nascent industries.
- Stage 2: Define the Protective Regime. Specify the breadth, depth, and length of the IP rights on offer. As a general rule, length should be easy to quantify; breadth and depth may not be. Nevertheless, it should be possible to devise fairly terse descriptions of the types of uses protected (depth) and of the degree of similarity necessary to constitute infringement (breadth).
- Stage 3: Calculate the Potential Return on Private Investment. Several determinations must be made. How will a private investor view the rights on offer? What up-front costs will be required to invest? How large will the returns have to be to justify the investment? How likely will they have to be? How much diversion is possible? When can the

unlikely to provide society with much additional innovation in exchange for the added costs inherent in lengthier protection.

investor expect to start seeing profits? How long can the profit stream be expected to continue?

- Stage 4: Consider the Societal Costs and Benefits. Several determinations must be made here as well. Could the desired investments be attracted with weaker rights (*i.e.*, does the proposed regime overprotect innovation)? Are insufficient rights deterring desirable investments (*i.e.*, does the regime underprotect innovation)?

It is important to note that although these stages are numbered, the analysis will rarely be sequential, and it is unlikely to ever follow a neat series of steps.[146] Nevertheless, these stages suggest a useful formalism for summarizing the nature, the costs, and the benefits of a proposed industry-specific reform of IP rights in line with the two goals underlying the framework's development: (i) the desire to highlight the differences among industries to reveal the legacy of the one-size-fits-all IP system; and (ii) the determination of whether a given proposal is likely to increase societal value. Stage 4 provides this determination. The framework's step-by-step approach insures that the Stage 4 answer is informed by an appropriate consideration of all relevant subsidiary issues, most notably the Stage 3 assessment of the private-sector's likely responses to the proposed reform.

B. *Transaction and Transition Costs*

IP regimes are neither developed nor implemented in a vacuum. Even a regime that appears to be theoretically optimal may prove to be unworkable, either because of the transaction costs associated with the regime's maintenance and enforcement, or because politically powerful entrenched interests will block any change to the *status quo*. While a full consideration of these concerns is beyond the scope of this article, no proposed framework for policy analysis can be considered complete without at least a brief discussion of transaction and transition costs.

Transaction costs refer to the costs of implementing a policy regime, making them an important part of a complete Stage 4 analysis. Some obvious examples of transaction costs associated with the current regime include the costs of running the Patent and Trademark Office and the Copyright office, the legal costs associated with the prosecution of patents, the litigation costs expended challenging or defending patents and copyrights, and the court costs generated by IP-related cases. Any proposed reform would be likely to require analogous categories of operating costs.

---

[146] The first two stages, for example, (the characterization of the regime and of the industry) are independent in some ways, and may be performed in either order. They are also interdependent in other ways, and may provide mutual feedback about the usefulness of various industry characterizations and protective specifications. The Stage 3 and Stage 4 considerations of private and public values share a similarly complicated interrelationship. In addition, issues raised during the evaluation of costs and benefits (Stages 3 and 4) will undoubtedly point to gaps in the characterizations (Stages 1 and 2), and may refine those analyses. Several iterations of each stage may be necessary to reach a coherent formulation.

A subtler form of transaction cost does not involve operations but rather recognizes that any industry affected by government regulation, including the awarding of IP rights, will hire lobbyists to protect its interests. These lobbyists play an important role in policy debates by keeping their clients' interests active and alive. At times, however, they can also distort the debate by forcing disproportionate attention on their clients' interests to the exclusion of the legitimate interests of other affected stakeholders. Lobbyists may thus simultaneously reduce the costs of collecting and assessing information sympathetic to their clients' viewpoints and increase the costs of collecting and assessing information that points in the opposite direction. When multiple powerful interests are implicated in different ways by a single reform proposal, their lobbyists may work against each other and thus absorb much of the cost of the policy debate. As the number of competing powerful interests declines, so does the government's ability to hear multiple perspectives without sinking substantial costs.[147]

The effect of reform on transaction costs may thus vary widely. Conservative proposals are likely to present transaction costs that are similar to those of the current regime. Radical proposals are likely to include both those that are much more expensive and those that are much less expensive than the current regime. Under either type of proposal, however, some reforms will reduce transaction costs while others will increase them. Net transaction costs must thus be assessed on a case-by-case basis.

Transition costs, on the other hand, are likely to be proportional to the degree of radicalism underlying a proposed regime change. The further the new regime is from the current system, the greater the expected transition costs. Transition costs arise whenever public policy changes the rules under which decision-makers must operate—and thus have implications to calculations of both Stages 3 (private) and Stage 4 (public) values. Consider but three simple examples of the challenges posed by *any* changes in the protections afforded to the innovative members of a given industry:

- A firm that invested in a product expecting to receive the existing protections may discover that its expectations were thwarted by the policy shift. Should the investing firm be compensated? If so, by whom, and how much? If not, to what extent does the uncertainty that would result from any high-level debate freeze investment?

---

[147] In some ways, this type of political economy argument may seem to cut against this article's primary thesis: IP rights applicable to many industries are likely to be subject to broad debate, while industry-tailored rights are more easily hijacked. In a broader sense, however, the recognition of this added potential cost is entirely consistent with the article's approach. Public policy reform is rarely easy. Even when the need for reform appears obvious to most, entrenched interests will attempt to direct public policy towards their own benefit—rather than towards that of society at large. Well-crafted reforms must include mechanisms for monitoring and evaluating the concerns of all potentially relevant stakeholder groups. This need for constant monitoring is an important transaction cost inherent in any policy regime.

- A firm that has already been awarded an IP right may not want to relinquish it in favor of the newly awarded rights. Might forcing it to do so constitute a taking? If the incumbent firm is allowed to keep its existing rights, but its competitors are only able to obtain the new types of rights, is the competitive playing field inherently uneven? Is there any way to develop fair competition in such an environment?
- Government specialists in the patent and copyright offices have been trained to assess applications for existing forms of IP protection. Attorneys and judges understand the law surrounding existing IP rights. Any regime shift will require retraining, re-education, and the development of new legal doctrine. The transition period may also be replete with uncertainty about the detailed workings of the new regime. This uncertainty, in turn, will lead to poor strategic decision-making, miscalculated investment decisions, and misallocated resources.

These concerns all argue in favor of conservative reforms. The closer the new rights are to the existing set, the smaller the necessary transition costs. It is important to note, however, that transition costs are only a small part of the story. Over the long run, costs sunk during a transitional period will be dwarfed by those incurred running the system. Furthermore, if the existing regime is inherently flawed, a cheap patch is unlikely to hold for very long, and transition costs are only likely to grow with time as expectations become increasingly entrenched.

Transition costs thus pose a particularly challenging problem to would-be reformers. On the one hand, analysts who *begin* by focusing on the political difficulties inherent in upsetting the status quo, on the disruption of expectation interests, on the possible takings consequences of changing rights already awarded, and on the inequity of grandfathering existing rights holders but not their direct competitors, will almost certainly conclude that conservative, incremental reforms are the only ones possible. On the other hand, commentators who ignore transition costs risk developing radically new, elegant systems of IP rights that have no chance of ever being adopted, and that are thus irrelevant to the real world.

Analyses of the transaction and transition costs associated with reform proposals should proceed in parallel with analyses of the merits of those proposals. This parallelism is likely to both weed out impractical proposals and incorporate early feedback from potentially affected parties, thereby smoothing eventual political acceptance. It is crucial to recall, however, that only proposals whose payoff net of changes in transaction costs and of transition costs exceed those of the current regime should be considered improvements.

All told, transaction costs and transition costs are an important part of any cost/benefit analysis. They need to be considered in both Stage 3 and Stage 4 of the analytic framework. With those caveats in mind, the essential framework is in place.

VI. THE SOFTWARE INDUSTRY

The preceding three sections were fairly theoretical. They began with an abstract return to first principles, added several layers of specificity, and progressed to a concrete analytic framework, albeit one stated in rather general terms. This section will present the first application of that framework to a specific industry: software. Section A will provide the Stage 1 industry profile. Section B will describe two potential protective regimes, as per Stage 2. Section C will assess private and public costs and benefits under each of these regimes, as required for Stages 3 and 4.

A. *Industry Basics*

Software firms engage in a kaleidoscopic set of concerns broadly focused on helping computer users achieve increasingly sophisticated tasks. The aspects of software, of software engineering, and of software management that appear central to the field's definition seem to change every few years. Fortunately, a full technical exposition of the industry is not needed to understand the ways that IP rights motivate software developers and shape the market. For present purposes, the answers to three key questions should provide the necessary industry definition:

- What types of software comprise the commercial industry?
- Which of the industry's characteristics drive consumer preferences?
- How can firms profit from developing software?

In addition to providing the information necessary for the Stage 1 industry profile, the answers to these questions will also help to shape the discussions of responsive protective rights (part of Stage 2) and of private sector rewards (part of Stage 3).

1. Platforms and Applications

Commercial software may be divided into two broad categories: platforms and applications.[148] The best-known examples of platforms are operating

---

[148] The Manifesto was strangely silent on this distinction. *See generally* Manifesto, *supra* note 11. Its view of the software industry appears to be monolithic, and essentially follows the analysis that the text applies to applications. Nevertheless, the platform/application distinction is real, and can have a profound impact on the analysis of incentives. To pick one obvious example, much of the behavior described in *Microsoft*-Facts was contingent on Microsoft's control of Windows, a platform program. *See Microsoft*-Facts, 61 F. Supp. 2d 1, 10-19 (D.D.C. 1999). Analogous behavior in the applications sector would have made little or no sense and would likely have been doomed to failure. The DC Circuit enshrined the platform/applications distinction as a matter of law by requiring a rule of reason analysis for tying claims involving platform software, while retaining a per se rule for all other software markets. *See Microsoft*-Appeal, 253 F.3d 34, 84-85, 95 (D.C. Cir. 2001).

This legal distinction, however, may provide litigants with less of a bright line than the Court believes, particularly when the issue involves allegations of tying. Many applications have served as quasi-platforms—at least during some stages of their development.

systems, such as DOS, Windows, OS/2, and Unix. Application programs include word processors, spreadsheets, and games.[149] Every usable computer system must consist of some underlying hardware, one platform program,[150] and at least one application. Each of these system components is responsible for a specific task. The hardware performs the computation that lies at the heart of all computing. Applications provide the functionality that the user desires. The platform allocates hardware resources among the applications and generally facilitates "communication" between the hardware and the application programs.[151]

This communication metaphor is critical to understanding software. Digital computers are machines capable of discriminating between high and low voltage levels.[152] Human computer users are far less adept at this task. They

---

Spreadsheets, for example, can frequently be augmented with "add-ins," separate software packages that enhance the functionality of the basic spreadsheet. Popular add-ins are frequently incorporated into later versions of the spreadsheet. From the perspective of an independent add-in producer, this incorporation could be described as a tie. The Court's ruling leaves open the question of the appropriate analytic regime to apply to such an allegation.

Its potential shortcomings as a legal bright line test notwithstanding, the platform/application distinction remains useful. A program that functions simultaneously as a platform and an application may force its developer to make an uncomfortable choice, particularly in a setting (such as the one described in the text) that tailors IP protection to the generally different marketing needs for the two classes of software. Nevertheless, this choice should rarely be *too* uncomfortable. These dual-use stages are generally transitory, and it is frequently clear towards which category the program is headed. Furthermore, as discussed in the text, platforms and applications occupy different niches in a virtual network and suggest different approaches to maximizing revenue. The distinction between them may thus become evident from the ways that they are marketed—as well as the ways in which they operate.

[149] *See* Definition of "Application Program" (last visited Nov. 28, 2001), *at* http://searchserviceprovider.techtarget.com/sDefinition/0,,sid28_gci507192,00.html ("An application program is any program designed to perform a specific function directly for the user or, in some cases, for another application program.").

[150] While it is possible to have more than one platform on a system, very few systems do. Two platforms *running* simultaneously are likely to conflict and make the system unusable. Even on systems with multiple resident platforms, only one may be used at any given time.

[151] *See Platform*, FOLDOC (Free On-Line Dictionary of Computing), *at* http://foldoc .doc.ic.ac.uk/foldoc/index.html (last visited Jan. 9, 2002), ("Specific computer hardware, as in the phrase 'platform-independent' may also refer to a specific combination of hardware and operating system and/or compiler, as in 'this program has been ported to several platforms.' It is also used to refer to support software for a particular activity, as in 'This program provides a platform for research into routing protocols.'").

[152] *See Voltage Levels and Signaling*, PCGUIDE.COM, *at* http://www.pcguide.com/intro /fun/clock.htm (last visited Jan. 9, 2002), ("While it is useful for us to think of data bits as being ones and zeros, this is in fact an abstraction. Within the PC's circuits, a one and a zero must be represented in some sort of physical manner. In fact, different components represent bits in totally different ways. On a hard disk, ones and zeros are encoded magnetically; on

prefer to describe the world in imprecise spoken languages, such as English. Computer scientists and computer engineers develop tools that allow users who speak English to communicate with machines that "speak" voltage levels. Large parts (if not all) of the work performed in these fields can thus be viewed as translation. Voltage levels are read, grouped, and coded into a sequence of languages that look increasingly like a specialized form of English.[153] This "upward" translation process eventually generates a platform program that allows the computer to function at a relatively high level.[154] At the same time, the user's English is restricted, formalized, modeled, and fed through specialized grammars to yield a sequence of languages that look increasingly like detailed engineering specifications.[155] This "downward" translation leads to an application program's input language.[156] When these two translation chains meet in a common language, communication between people and computers becomes possible.

While a full translation chain was always necessary for people to use computers, the locus of technical and commercial attention has shifted as computers have matured. In the 1960s, virtually all computer users were technically trained professionals who were personally proficient in Fortran, COBOL, ALGOL, LISP, or some other specialized language that bore only a cosmetic relationship to English.[157] By the 1990s, many accomplished computer users knew no machine language more technical than the Boolean inputs to a search engine or a set of point-and-click instructions.[158] This shift was enabled by computer engineers who developed technologies that increased hardware power and by computer scientists and programmers who availed themselves of that power to advance the translation chain built up from voltage levels progressively closer to English.[159]

The technological innovations that shifted the machine's "understanding" closer to English also shifted the balance between platforms and applications—as well as the balance between hardware and software. Each technological generation incorporated more of the tasks that had once been required to facilitate translation down from English into the chain that grew upward from voltage levels. These shifts allowed tasks to migrate from applications to

---

an optical disk, by a sequence of pits and lands. And within the core operating circuitry of the PC, ones and zeros are represented by *voltage levels*.") (emphasis in the original).

[153] *See generally*, IVAN FLORES, COMPUTER PROGRAMMING (Prentice Hall 1966) (providing a background of various computer programming principles).

[154] *See id*.

[155] *See id*.

[156] *See id*.

[157] *See The Computer Museum History Center*, *at* http://www.computerhistory.org (last visited Dec. 5, 2001) (providing a detailed chronology of the history of computing); *see also Charles Babbage Institute*, *at* http://www.cbi.edu (last visited Dec. 5, 2001) (providing resources on the history of computing).

[158] *See The Computer Museum History Center*, *supra* note 157.

[159] *See id*.

platforms to hardware and allowed software developers to focus on the development of input languages and interfaces that appeared increasingly natural to human users. As a result, the boundaries between platforms and applications have shifted with every generation of technology. Commercial opportunities have followed that technological lead.

Under any generation of the technology, successful communication between the platform and the applications is critical; without this last remaining translation, the entire system is useless. Because platforms typically communicate with many applications while applications only communicate with one (or at most, a small number of) platforms, the sensible approach to this final translation is for the applications to "learn" the platform's language, rather than the other way around. Platform developers facilitate this education by publishing dictionaries and grammars that teach potential application developers how to communicate with their platforms. In contemporary parlance, these translation aides are known as "application programming interfaces" ("APIs").[160]

2. Software as a Network Industry

Despite the consistent (and ongoing) changes that the software industry encounters as technology advances, some features of the industry have remained invariant:

- The number of platform programs circulating approximates the number of computers (*i.e.*, virtually all computers house exactly one platform).
- There are many application programs on each computer.
- *Special-purpose* users require a specific application. These users *must* select a platform that can support that application.[161]
- *General-purpose* users want to own a functioning computer system that can accomplish a broad range of tasks; their concern with the specifics of those tasks is secondary. These users thus tend to select a platform and accept the *de facto* restriction to applications that can communicate with the selected platform.
- In every technological generation to date, the general-purpose market has dwarfed the special purpose market.
- General-purpose application developers want to attract as many users as possible. They will thus choose to develop applications that communicate with as many viable platforms as possible, starting with the most popular. The number of users whose platform can

---

[160] Although "API" is a relatively recent term, the role of the API has always been present in computer technology. For the purposes of this article, the term API will refer to a basic functional role, whether or not the term is technically applicable to the generation of technology being discussed. *See Microsoft*-Facts, 65 F. Supp. 2d 1, 9, 13, 19-22 (D.D.C. 1999) (providing some basic definitions and discussing the importance of APIs).

[161] Consider, say, a movie studio purchasing a specialized state-of-the-art animation package to facilitate special effects. For these purchases, the availability of the application is likely to dictate all other purchases, including both platform and hardware.

communicate with an application defines the maximum market for that application.

- General-purpose platform developers want to attract as many users as possible to their platforms. Because platform choice is guided, at least in part, by the availability of applications, platform developers will want to see a large suite of applications compatible with their platforms.

Some of these observations have been controversial at times and were once less obvious than they appear to be today. For the remainder of this discussion of software industry incentives—which will focus entirely on the general-purpose software market—they will be taken for granted.

Because many of these intergenerational invariant characteristics flow from software's classification as a *network industry*,[162] a brief overview of the economics of network industries is a prerequisite to understanding the incentives that motivate software developers. Network industries, as their name implies, have a fair amount in common with physical networks. Physical networks, such as the telephone or the electricity networks, are basically collections of end users (*i.e.*, people who use telephones or electrical appliances) connected by a physical link, such as a wire. Any two items wired to the same network must be *interoperable*, or able to conform to the specifications of the network. Anyone who has ever attempted to plug an American appliance into a European electrical outlet has experienced the frustration of non-interoperable equipment; a perfectly good appliance plus an equally good but incompatible outlet equals nothing of value.

Not all networks, however, require physical wires. Users of interoperable software define a *virtual* network. Examples of software networks are the collections of all people using software that can run in Windows or in Linux. These people are "connected" by the APIs to which all application developers, and thus all users, must conform. Entire industries can also be examples of virtual networks. One important defining characteristic of a network industry is "positive feedback," whereby the value of a network grows with the size of the network.

Consider again the example of appliances and electrical networks. An American appliance cannot be plugged into a European outlet without an adapter. An American appliance with an adapter can be plugged into a European outlet, but it will quickly burn out unless it is also attached to a converter. Imagine a consumer attempting to purchase an appliance in an environment of multiple competing electrical systems—a situation that actually existed in the U.S. during the earliest days of electricity. A logical consumer would select the appliance best suited to her home. But what if her

---

[162] *See generally*, CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES, 173-225 (Harvard Business School Press 1999) (providing an excellent non-technical introduction to the economics of network industries). *See also* Mark A. Lemley and David McGowan, *Legal Implications of Network Economic Effects*, 86 Calif. L. Rev. 479 (1998) (discussing the ways in which network economics have already had an impact on several areas of the law, including IP law).

home were not yet wired for electricity? If that were so, she would face two decisions: choosing both an appliance and an electrical network. Again, being logical, she would probably recognize that the widest selection of appliances existed for the most popular electrical network and thus choose to join that largest network. Her rational, value-driven choice would thus make the largest network even larger.

This phenomenon also exists in the software world. Application developers would like to access the broadest possible market. If more people own computers with operating system W than with operating system U, then W-compatible applications will be written for a larger market than U-compatible applications. Application designers will rationally decide that, all else being equal, they would prefer to sell into the larger market. These decisions will add to the collection of W-compatible software, thereby making W even more attractive to new consumers. This cycle illustrates the positive feedback phenomenon.

By way of summary, network industries have two basic characteristics relevant to understanding the incentives of industry participants:[163]

- All components of the network must be mutually compatible (*i.e.*, interoperable); and
- The value of the network increases with the number of members, users, or subscribers (*i.e.*, positive feedback).

3. Paths to Profitability

These characteristics point toward a strategy for generating revenue that is unique to network industries. The importance of positive feedback suggests that firms should disseminate the components that *define* their networks as widely as possible and at the lowest price possible—zero, if necessary. Once consumers are locked into the network,[164] the firm that holds the network's

---

[163] Interoperability and positive feedback are not the only important characteristics of network industries. They are, however, the two most relevant to the motivation of innovation. For commercial opportunities, lock-in, discussed in *infra* note 164, is at least as important.

[164] Lock-in is a third defining feature of true network industries. If consumers are not locked into the network owner, any steps that the network owner tries to take to increase his profits above the competitive level will cause consumers to shift their business to competing networks. Even a firm owning a monopoly network may be susceptible to such switching if entry is easy, uncommitted entrants exist, or consumers can integrate backward to take the item being provided in-house. The relationship between locked in consumers and anticompetitive behavior was first formalized as a matter of antitrust law in *Eastman Kodak Co. v. Image Technical Services* 504 U.S. 451 (1992). For a discussion of the ways in which the recognition of lock-in affected antitrust analysis, *see generally*, Bruce D. Abramson, *Analyzing Antitrust Analysis: The Roles of Fact and Economic Theory in Summary Judgment Adjudication*, 94 ANTITRUST. L. REV. 303 (2001). The relationship among anticompetitive behavior, network economics, and lock-in played an important role in *Microsoft*, where the Court recognized that much of Microsoft's anticompetitive behavior

keys can then profit through a variety of interoperable add-on sales and services or by selling network access to its competitors.

Consider, for example, the strategies available to a firm that manufactures operating systems. If operating systems did not exhibit network externalities, the task would be straightforward (although not necessarily easy): The firm would establish a profit maximizing price for its operating system and attempt to sell as many copies as possible at that price. Because operating systems define networks, however, other strategies may be more appropriate. The firm could (and should) recognize that it can generate revenues by selling applications that run on its operating system, by licensing its APIs to competitors who promise to develop applications, by selling tools and/or development kits to would-be application developers, by providing after-sales service, or by some combination of the four.

This example highlights the differences between the incentives of platform developers and the incentives of application developers. In terms of a Stage 1 industry analysis, then, software actually breaks down into two industries with distinct incentive patterns. The most likely source of revenue to a platform developer comes from the sale of network access. The most likely source of revenue to an application developer comes from the sale of software. A first-principles approach to IP rights must assess the likely impact of these differing incentive structures on the industry's participants.

B. *Alternative Regimes for the Protection of Software*

Perhaps the simplest way to approach the analysis of incentives is to consider the ways in which they might guide behavior under different protective regimes. A comparison of two regimes should suffice to raise most relevant issues. This section thus describes how the current regime works, considers some signs of existing underprotection and overprotection, and then contrasts it with a proposed reform based on the principles outlined in the Manifesto.[165]

To reiterate software's challenge to the IP system, computer programs are textual works designed to generate functional behavior.[166] They deliver that functionality by cloaking a comprehensible textual work (of the sort typically circulated and protected by copyright) in an incomprehensible work of "machine language." Software is thus a complex product that exists on many levels, at least three of which have received legal protection and the scrutiny of the courts:

- *Source Code* is the collection of algorithms written by computer programmers in a programming language, such as Fortran, Pascal,

---

was enabled by the "applications barrier to entry," a phenomenon that the court described in almost textbook network-industry terms. *Microsoft*-Facts, 65 F. Supp. 2d at 28-36.

[165] *See generally* Manifesto, *supra* note 11.

[166] *See* discussion in § I setting out the challenge presented by software to the IP system.

LISP, C++, or Java.[167] Source code is readable and comprehensible by trained professionals.[168] Computer programmers or engineers given access to source code should be able to understand a program's functionality, and to replicate that functionality in their own work.[169]

- *Object Code* is a pre-compiled version of the source code that translates the programming language into something closer to machine language.[170] Object code is generally incomprehensible to any human reader, regardless of training; it is intended for use by machines only.[171] Access to object code generally allows users to *use* the software, but not to decipher the underlying functional mechanisms.[172]
- *The User Interface* defines the way in which a user experiences the software.[173] The "look and feel" of text and objects on a user's monitor

---

[167] *See Source Code*, WHATIS?COM, *at* http://searchwebmanagement.techtarget.com/sDefinition/0,,sid27_gci213030,00.html (last visited Jan. 9, 2002), ("Source code and object code refer to the 'before' and 'after' versions of a computer program that is compiled . . . before it is ready to run in a computer. The source code consists of the programming statements that are created by a programmer with a text editor or a visual programming tool and then saved in a file. For example, a programmer using the C language types in a desired sequence of C language statements using a text editor and then saves them as a named file. This file is said to contain the source code. It is now ready to be compiled with a C compiler and the resulting output, the compiled file, is often referred to as object code. The object code file contains a sequence of instructions that the processor can understand but that is difficult for a human to read or modify. . . .When you purchase or receive operating system or application software, it is usually in the form of compiled object code and the source code is not included.")

[168] *See id*.

[169] *See id*.

[170] *See id*.

[171] *See id*.

[172] *See id*.

[173] *See User Interface*, WHATIS?COM, *at* http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214505,00.html (last visited Jan. 9, 2002).

> In information technology, the user interface (UI) is everything designed into an information device with which a human being may interact—including display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it. In early computers, there was very little user interface except for a few buttons at an operator's console. The user interface was largely in the form of punched card input and report output. Later, a user was provided the ability to interact with a computer online and the user interface was a nearly blank display screen with a command line, a keyboard, and a set of commands and computer responses that were exchanged. This command line interface led to one in which menus (list of choices written in text) predominated. And, finally, the graphical user interface (GUI) arrived, originating mainly in Xerox's Palo Alto Research Center, adopted and enhanced by Apple Computer, and finally effectively standardized by Microsoft in its Windows operating systems. The user interface can arguably include the total 'user experience,' which may include the aesthetic appearance of the device, response time, and the content that is presented to the user within the context of the user interface.

> define the interface.[174] For most computer users and commercial software purchasers, the interface and its underlying capabilities *define* the relevant product.

Any IP regime designed to protect software must address at least the first two of these levels. Protection of the interface, while important, is an orthogonal concern. Because source code and object code capture the functional components of the software, they are central to both the nature of the product and the challenge posed by functional copyrights. Interface issues actually arise in only a small (albeit highly visible) portion of programs, and implicate a fundamentally different set of concerns—namely those related to look, feel, layout, and artistic expression.[175] The analysis will thus touch upon interface issues only tangentially.

1. The Current Regime

Although the current IP regime protecting software incorporates aspects of patent, copyright, and trade secret law,[176] copyright remains the most important protection granted to software developers. It is not difficult to see the appeal of software copyrights (at least to the legal and administrative communities charged with granting and enforcing the protection). Their pitfalls are subtle and complex—but potentially damaging nonetheless.[177]

The most obvious appeal of granting copyright protection to software is that software *looks like* something that should be copyrighted; it is comprised of words, drawn from a limited lexicon, and combined to conform to the rules of a grammar. Multiple copies can be produced at low cost and circulated at the discretion of the copier. In most cases, copyright law affords the copyright holder the exclusive right to reproduce, to distribute copies, or to perform the original,[178] as well as fairly broad rights in "derivative works."[179] Software also controls the arrangement of images and/or words on a video screen. Again, such arrangements have long been protected by copyright, and there is

---

Id.

[174] *See id*.

[175] *See, e.g.*, Lotus Dev. Corp. v. Paperback International, Inc., 740 F. Supp. 37 (D. Mass. 1990); Apple Computer, Inc. v. Microsoft Corp. 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd* 35 F.3d 1435 (9th Cir. 1994).

[176] This integration of three bodies of law has led to a fairly complex web of simultaneous IP rights. This point is discussed further in *infra* note 200.

[177] Professor Karjala has provided several detailed discussions of the difficulties inherent in the decision to protect software with copyrights. Needless to say, the points raised here are only the tip of the iceberg. For more detailed analyses, *see, e.g.*, Dennis S. Karjala, *The Relative Roles of Patent and Copyright in the Protection of Computer Programs*, 17 MARSHALL J. COMPUTER & INFO. L. 41, 50-6 (1998); Dennis S. Karjala, *Copyright Protection of Operating Software, Copyright Misuse, and Antitrust*, 9 CORNELL J. L. PUB. POL'Y 161, 171-82 (1999).

[178] *See* 17 U.S.C. § 106 (2000).

[179] *See* 17 U.S.C. § 103.

no obvious reason that arrangements generated by software should be any different from their more conventional counterparts.[180] The extension of copyright protection to software thus appears to be a natural fit—at least at first blush.

Software, however, differs from traditional copyrightable works in (at least) one crucial way: it is *functional*. Congress developed copyright law to protect "artistic" expressions—or, more plainly, expressions of ideas rather than the ideas themselves.[181] Although even traditional texts and artworks can create perplexing problems at the margins, the distinction between an idea and its specific, copyrightable expression is usually evident. When copyright law began extending its reach into compilations and organizations, issues related to functionality began to emerge.[182] Even in those areas, the expression captured in the copyrighted work, *not* the underlying functional concept, benefited from protection. In many ways, then, the use of copyright protection for an inherently functional product has the feel of fitting a "square peg into a round hole."[183]

This misfit has led some courts to develop a nascent doctrine of copyright misuse, which prohibits a copyright holder found liable of attempting to extend his limited monopoly beyond the allowable confines of copyright law from enforcing his copyright.[184] This doctrine, borrowed from patent law, was found (for the most part) in settings related to patents granted on goods or processes important as inputs to other goods.[185] It is unclear how the doctrine

---

[180] The "look and feel" cases dealt with precisely this issue—the "feel" conveyed by the shape and arrangement of icons on a desktop. *See, e.g.*, *Lotus Dev. Corp. v. Paperback International, Inc.*, 740 F. Supp. at 62-65; *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d at 1442-45.

[181] *See* NIMMER & NIMMER, *supra* note 121, § 2.03[D] (2001).

[182] *See id.* § 2.18.

[183] Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1992).

[184] *See* Lasercomb v. Reynolds, 911 F.2d 970, 973 (4th Cir. 1990).

[185] Copyright misuse draws liberally on the accepted doctrine of patent misuse. In a copyright misuse case, a copyright holder agrees to license use rights to a second party—but only subject to a variety of restrictive conditions. In *Lasercomb*, 911 F.2d 970, for example, Lasercomb developed Interact, a copyrighted CAD/CAM package that aided the creation of steel rule dies used in the manufacture of cardboard boxes. Lasercomb licensed a fixed number of copies of Interact to Reynolds. Lasercomb's standard Interact licenses also contained a clause that prohibited Interact licensees from creating a directly competitive software package. Reynolds made several unauthorized copies of the program, and Lasercomb sued for copyright infringement. Reynolds defended, *inter alia*, by claiming copyright misuse. In assessing this defense, the Court noted that "much uncertainty engulfs the 'misuse of copyright' defense. We are persuaded, however, that a misuse of copyright defense is inherent in the law of copyright just as a misuse of patent defense is inherent in patent law." *Id.* at 973. The Court then concluded that Lasercomb's license term *did* constitute a misuse of its copyright. Other courts have reached similar conclusions about the validity of the copyright misuse defense, notably the Ninth Circuit in Practice Mgmt. Information Corp. v. American Med. Ass'n, 121 F.3d 516, (9th Cir. 1997) (the AMA's

would apply to non-functional goods—a vagueness that explains (at least in part) why a comparable copyright law doctrine did not seem necessary prior to the advent of functional copyrights.

Functional copyrights pose yet another challenge to the courts. One of the classic boundaries between works that can be copyrighted and those that cannot be lies in the idea/expression distinction.[186] Copyright protection extends only to the expression of ideas, not to the ideas themselves.[187] With respect to functional works, the line between an idea and its expression can be less than obvious. This difficulty has been recognized since at least 1879, when the Supreme Court ruled that the general approach to bookkeeping implicit in a copyrighted accounting ledger was a "method of operation," and thus an idea, not an expression.[188] In more recent years, this distinction has produced a circuit split with respect to software interfaces.[189] After all, *every* interface embodies a method of operation and hence an idea. At the same time, every interface also embodies a set of discretionary decisions about key words, icons, layouts, etc. and therefore embodies an expression.

The challenge posed by software, however, goes far beyond the interface. The very nature of a computer program blurs the distinction between an idea and its expression and thus makes it quite difficult to determine when a second program that shares both functional and layout characteristics with the first has infringed. Various courts have proposed elegant tests for determining when a second program is "close enough" to an earlier copyright-protected competitor to constitute infringement. Perhaps the most influential of these tests is the abstraction-filtration-comparison test, first introduced by the Second Circuit in *Computer Assoc. Int'l v. Altai*[190] and subsequently adopted by several other circuits.[191] Under this test, courts consider protectable and unprotectable

---

licensing of a coding system for medical procedures on the condition that a licensee not develop a competing code constituted copyright misuse), and the Fifth Circuit in Alcatel, Inc. v. DGI Technologies Inc., 166 F.3d 772 (5th Cir. 1999) (Alcatel's licensing term requiring that its copyrighted software be used only in conjunction with Alcatel equipment—and not competing equipment—constituted copyright misuse). Despite the growing number of lower courts that have come to recognize the copyright misuse defense—and despite the growing relevance of the defense in a society with increasing numbers of functional innovations protected by copyright rather than by patent—the Supreme Court has yet to consider the doctrine.

[186] *See* NIMMER & NIMMER, *supra* note 121, at § 2.03[D].

[187] *See id.*

[188] *See* Baker v. Selden, 101 U.S. 99, 103-5 (1879).

[189] *Cf.* Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by equally divided Court*, 516 U.S. 233 (1996) (the keystrokes necessary to operate Lotus's spreadsheet program constitute a method of operation, and are thus unprotectable by copyright) *with* Autoskill, Inc. v. Nat'l Educ. Support Sys., Inc., 994 F.2d 1476, 1499 (10th Cir. 1993) (the keying procedure on Autoskill's program can be protected by copyright).

[190] 982 F.2d 693 (2d Cir. 1992).

[191] *See Lotus Dev. Corp. v. Borland Int'l*, 49 F.3d at 814-15; Comprehensive

program characteristics separately, and find infringement only if the second program is too similar to the protected components of the first.[192]

The Ninth Circuit provided a rather high profile example of this test when it considered Apple's claim that early versions of Microsoft Windows infringed its copyrighted operating system and interface for the Macintosh.[193] The court noted that all similarities between the two programs must have come from one of three sources: (i) an earlier license granted by Apple; (ii) obvious expressive interpretations of the same underlying idea; and (iii) outright copying.[194] The first two sources were legitimate; the last was a potential source of copyright infringement.[195] The court then identified all similarities (abstraction), traced most of them to one of the first two categories (filtration), and compared what remained as potential infringement (comparison).[196] The court ruled that Apple failed the comparison test and thus affirmed the lower court's finding that Microsoft had not infringed Apple's copyright.[197]

The *Altai* test notwithstanding, no bright line separates the protected *behavioral* elements of a program from its unprotected elements. While courts regard both source code and object code as literal elements of a creation, and thus as protected by copyright law,[198] this protection provides little comfort to software developers whose primary interest lies in protecting their programs' behavior—not its literal expressions of that behavior.[199]

As things stand, many (if not most) commercial software firms do not rely on copyright law alone to protect their source code. Instead, they maintain their source code as a trade secret while obtaining copyright protection on their object code.[200] This duality provides software developers with a unique form

---

Technologies Int'l. v. Software Artisans, 3 F.3d 730, 735 (4th Cir. 1993); Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 834 (10th Cir. 1993); Kepner-Tregoe, Inc. v. Leadership Software, 12 F.3d 527, 534 (5th Cir. 1994); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1544 (11th Cir. 1996); Control Data Sys. v. Infoware, Inc., 903 F. Supp. 1316, 1322-24 (D. Minn. 1995).

[192] *See Bateman*, 79 F.3d at 1544.

[193] *See* Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994).

[194] *See id.* at 1443, 1447.

[195] *See id.*

[196] *See id.*

[197] *See id.* at 1447.

[198] "It is now well settled that the literal elements of computer programs, *i.e.*, their source and object codes, are the subject of copyright protection." *Altai* 982 F.2d at 702 (citing Whelan Assoc., Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1233 (3rd Cir. 1986)) (source and object code); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d. Cir. 1983) (source and object code); Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870, 876-77 (3rd Cir. 1982) (object code).

[199] *See* Manifesto, *supra* note 11, at 2310-32.

[200] This copyright/trade secret duality oversimplifies the current use of IP rights by the software industry in at least two ways. First, it omits the distinct but related problems caused by software patents and by software protected simultaneously by patent and

of protection. When an author circulates her conventional copyrighted work, anyone who comes into legitimate possession of a copy of the work has legal authorization to do three things:

- Read (or otherwise use) the copyrighted work.[201]
- Understand the idea underlying the copyrighted work and create an independent expression of that idea.[202]
- Make some class of "fair uses" of the copyrighted work.[203]

A software firm's ability to circulate only object code allows it to split the legal rights that accompany a legal copy of the work. In limiting circulation of the program to object code, the holder of a software copyright, unlike

---

copyright law. Second, it does not mention that source code is typically copyrighted along with object code. Neither of these simplifications detracts from the text's basic assertion that the current paradigm overprotects software—in fact, they both tend to strengthen the argument.

In terms of software patents, it is important to recall that patents protect only a relatively small percentage of commercially viable software products. When present in a complex software package, they extend the protection even further than the "standard" combination of copyright and trade secret. A patented algorithm embedded in copyrighted object code derived from trade secret protected source code is extraordinarily well protected. It represents a mathematical formula captured by a specific representation that may not be circulated freely. Reverse engineering may reveal complicated interactions between patented and unpatented components of the software. While such circumstances strengthen the overprotection analysis described in the text, they are not necessary for software to be overprotected. The combination of copyright and trade secret protection suffices—with or without the complications added by patents.

A discussion of source code copyrights is similarly unnecessary to demonstrate overprotection. While software firms may copyright their source code, they rely on trade secret law for most of its protection. In fact, they typically copyright their source code without ever revealing its contents. The Copyright Office accommodates this dual protection by allowing authors of software containing trade secrets to black out the trade-secret protected portions in the deposited copies of their source code. Furthermore, if the author is unwilling to deposit source code, the Office will register the software under its rule of doubt. *See* U.S. COPYRIGHT OFFICE, CIRCULAR 61: COPYRIGHT REGISTRATION FOR COMPUTER PROGRAMS 2, *available at* http://www.loc.gov/copyright/circs/circ61.pdf (last visited Jan. 9, 2002). The industry's primary reliance on trade secret protection has become increasingly evident as both Congress and the courts have come to permit various forms of decompilation—the one practical way of defeating a trade secret against which a standard application of copyright principles could have defended. *See, e.g.*, 17 U.S.C. § 1201(f) (2000) (allowing decompilation for the purposes of achieving interoperability among computer programs); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1992) (allowing Accolade to reverse engineer Sega's game programs in order to create games compatible with Sega's hardware). The key to the overprotective paradigm described in the text is thus *not* that different protections are sought on different aspects of the software, but rather that different protections are *used*.

[201] *See* NIMMER & NIMMER, *supra* note 121, at § 13.01.

[202] *See id.*

[203] *See id.* at § 13.05.

copyright holders of other subject matter, can permit customers to *use* her work while prohibiting them from understanding it, integrating it into their own world views, and creating an independent expression of the same idea. This ability necessarily turns the software copyright into a much more powerful and valuable form of protection than it ever was in a conventional, textual setting.[204]

Copyright law was not designed to cope with this type of situation. Functionality was essentially an afterthought in copyright law. In contrast, patent law, developed to protect functional innovations, contains a very powerful combination of responsive mechanisms: mandatory disclosure, blocking patents, and cross licensing.[205] Copyright law contains no analogous provisions.[206]

The Stage 2 parameterization of the current regime may thus be stated as follows: Existing software copyrights are narrow, shallow, and long. They are narrow because behavior is not protected by copyright, and behavior is the component of the innovation that the copyright holder would most like to protect.[207] They may be shallower than a standard copyright because of the courts' apparent willingness to allow decompilation by commercial competitors as part of the reverse engineering process.[208] Their length is effectively infinite because copyright protection lasts far longer than the useful

---

[204] This dual desire to own a widely distributed product while retaining control over a competitor's access is common to all participants in network industries. Firms in most industries must choose between wide distribution and controlled access whereas software firms need not make such a choice. The dual nature of their product allows them to keep source code secret (thereby controlling access) while simultaneously distributing object code (thereby gaining wide distribution). In 1998, Congress strengthened the rights afforded by copyright law to software designers wishing to take advantage of this dual protection by enacting the DMCA 17 U.S.C. § 1201(a)(1)(A), prohibiting the "circumvent[ion of] a technological measure that effectively controls access to a work protected under this title [the Copyright Act]." At the same time, however, the DMCA also explicitly allowed various forms of reverse engineering of a computer program in which a user effects the circumvention "for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program. . ." 17 U.S.C. § 1201(f)(1).

[205] *See* CHISUM ON PATENTS, *supra* note 18, § 7.01 et seq., 16.02, 19.01 et seq.

[206] *See id.* at § 14.01 et seq.

[207] This desire differs from that of an innovator who produces a typical textual or artistic work protected by copyright. Those innovators are frequently more interested in protecting their expression than the underlying knowledge that it embodies.

[208] Even courts that have allowed decompilation have recognized that their rulings are part of a growing trend of anomalies in software copyright law. *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1992). In any event, the effect of the legal treatment of reverse engineering on protective depth remains uncertain—it might make the protection either deeper or shallower than conventional copyrights depending on the ultimate determination of permissible actions to reverse engineer without infringing.

life of computer code. Existing copyrights by themselves thus appear to underprotect the rights of software developers.

Software is anomalous because innovators do not have to opt out of trade secret protection to acquire IP rights. The combined copyright/trade secret protection is broader and possibly deeper than conventional (*i.e.*, non-software) copyrights, and is of effectively infinite length. Programmers other than the original developer are blocked from incorporating behavior, expression, and a full range of activities that require access to the knowledge embedded in the source code. The *de facto* protection of behavior thus makes software copyrights broader than their conventional counterparts. The prohibition against some forms of decompilation and reverse engineering may make them deeper, as well.[209] This dual protection thus leads to a situation in which current IP rights *plus* trade secret protection are likely to be overprotective.

2. The Manifesto Proposal

The Manifesto recognized many of these shortcomings of the current regime.[210] In particular, the Manifesto saw three aspects of the current regime as particularly damaging to the long-term health of the software industry: a misplaced balance between the protection of a program's literal and behavioral elements;[211] insufficient attention paid to reverse engineering;[212] and the ability of a software developer to gain IP rights without disclosing the inner workings of his innovation.[213] These difficulties are necessarily intertwined.

The first key difficulty requires understanding the nature of a computer program. To most users, the program appears through its interface, the collection of icons and text revealed on the monitor. These icons and arrangements are literal elements of the program and are thus protected by copyright law.[214] Software developers care about this protection, and often litigate perceived infringement. Interfaces are only interesting, however, if they connect the user to a functioning program that accomplishes the tasks required of it. These necessary functional components of the program define behaviors. They embody the algorithms devised by the software developer. For the most part, innovative software embodies innovative algorithms. In order to work, the algorithms must be translated from the quasi-mathematical

---

[209] Although, as noted in *supra* note 200, the prohibition on decompilation has been weakened considerably by both case law and statute, and may now represent only a minimal depth increase. Again, as noted in *supra* note 208, the effect of the legal treatment of reverse engineering on protective depth remains uncertain.

[210] *See generally*, Manifesto, *supra* note 11.

[211] *See id*. at 2312-13.

[212] *See id*. at 2336-38.

[213] *See id*. at 2336-40.

[214] *See, e.g.*, Lotus Dev. Corp. v. Paperback International, Inc., 740 F. Supp. 34, 65-68 (D. Mass. 1990); Apple Computer Inc. v. Microsoft Corp., 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd* 35 F.3d 1435 (9th Cir. 1994).

language in which they were first expressed into a programming language that the machine can understand. Under standard copyright rules, these programming language translations (as well as their further translations into object code) are protected as literal elements of the work, or as expressions of the algorithm. The underlying mathematical algorithm remains an idea—not an expression—and can thus not be protected by copyright law.[215] This situation thus underprotects the innovations that society would most like to motivate.

This mismatch between the reward system and the types of innovation that it purports to motivate leads to the second key deficiency of the current regime: the inappropriate attention paid to reverse engineering. The reverse engineering of a copyrighted work is usually trivial, as the work itself is facially obvious. In copyright-protected software, however, the "masking" of the facially obvious source code behind the opaque object code renders reverse engineering difficult—but not impossible. In particular, software developers have found two ways to reverse engineer competing products.

The first route employs "black box" testing.[216] A software engineer employing this technique devises a series of test inputs, observes the performance of a program (to which he has only object code access) under each of the inputs, and then works backward to infer the underlying source code. The Manifesto views this technique as powerful (and often successful), and thus sees it as a genuine threat to a software developer's ability to protect the behavioral aspects of her programs. As an empirical matter, it is not clear how useful black box testing is to programmers attempting to decode complex competing programs. Black box testing certainly works well on gross behavior. Intricacies of interoperability and robustness under unusual input sequences, however, are much harder to duplicate.

Black box testing thus appears to be a greater threat to truly novel and revolutionary programs than to incremental advances over the state of the art. These novel programs, however, are also those most likely to be protected by patents, and software patents provide greater protection against reverse engineering than do software copyrights. This greater protection is not without its own pitfalls. Cohen and Lemley argued that the standard protection against reverse engineering inherent in patent law might be strong enough to impede the development of next-generation patented software.[217] This concern inverts the Manifesto's argument that the holders of software copyrights are underprotected from reverse engineers.

The second route to reverse engineering lies through decompilation. A decompiler is a computer program whose purpose is complementary to that of a compiler. While a compiler translates human-readable source code into

---

[215] *See* NIMMER & NIMMER, *supra* note 121, at § 2.03[D].

[216] *See* Manifesto, *supra* note 11, at 2317.

[217] *See* Cohen and Lemley, *supra* note 73, at 5.

machine-readable object code, a decompiler does the opposite.[218] Thus, a competitor who purchases object code and decompiles it successfully also possesses a copy of the copyright-protected source code—by definition, an unauthorized copy.

The status of reverse engineering through decompilation appeared to be controversial prior to the passage of the DMCA—which appears to allow it.[219] In one well known pre-DMCA case that addressed this question, Accolade purchased Sega's video game programs, decompiled them, learned how they worked, and used that knowledge to build competing games.[220] Sega sued for copyright infringement because of the decompilation.[221] Accolade contended that its creation of an unauthorized copy for purposes of reverse engineering a trade secret qualified as fair use.[222] The court ruled in favor of Accolade, but noted that its decision appeared incongruous with copyright law doctrines developed in more traditional domains.[223] The court explicitly labeled this decision as but one more of a growing trend of anomalies in software copyright

---

[218] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1515 n.2 (9th Cir. 1992) (describing decompilers).

[219] The DMCA prohibited the circumvention of anti-copying devices, *see supra* note 204, but explicitly allowed various forms of reverse engineering of a computer program in which a user effects the circumvention "for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program. . . ." 17 U.S.C. § 1201(f)(1) (2000). The one court to have directly addressed the DMCA's prohibition on circumvention ruled that the use of a decryption algorithm to allow the unauthorized copying of DVDs "clearly is a means of circumventing a technological access control measure," and thus violates the prohibition of § 1201(a). Universal City Studios, Inc. v. Reimerdes, 111 F. Supp. 2d 294, 317 (S.D.N.Y. 2000) *aff'd sub. nom.* Universal City Studios v. Corley, 2001 U.S. App. LEXIS 25330 (2d Cir. 2001). The status of decompilation under these new provisions had not been addressed in any case reported before the end of 2001. To further complicate matters, there does seem to be an understanding that "section 1201 of the DMCA occupies 'a niche distinct from copyright infringement' and that section 1201 is removed from the [Copyright] Act's definition of copyright infringement." RealNetworks, Inc. v. Streambox, Inc., 2000 U.S. Dist. LEXIS 1889 (W.D. Wash, 2000) (quoting NIMMER & NIMMER, *supra* note 121, § 12.A17[B] (1999 Supp.)). Under current law, then, decompilation appears to be an uneasy exception to general copyright tenets, recognized by some courts, whose status under the DMCA has yet to be adjudicated fully. Furthermore, the DMCA—as a copyright act—says nothing at all about the decompilation of patented software.

[220] *See Sega Enters. Ltd.*, 977 F.2d at 1527. *See also* Atari Games Corp. v. Nintendo of America, Inc. 975 F.2d 832 (Fed. Cir. 1992) (similarly holding that decompilation did not constitute infringement).

[221] *See Sega Enters. Ltd.*, 977 F.2d at 1515-16.

[222] *See id.* at 1520.

[223] *See id.* at 1527. Under current law, then, decompilation is an uneasy exception to general copyright tenets, recognized by some courts, whose status under the DMCA has yet to be adjudicated fully.

law that has emerged from the tendency to (or the explicit attempt to avoid) forcing "the proverbial square peg into a round hole."[224]

The DMCA's apparent allowance of decompilation as a form of reverse engineering[225] opens a potentially powerful route for competitors wishing to defeat trade secret protection. Various commentators have described decompilation as practical and significant.[226] The Manifesto described its apparent absence from copyright law as a potential source of underprotection,[227] while Cohen and Lemley cautioned that its apparent presence in patent law is a potential source of overprotection.[228] The true significance of decompilation remains to be seen. While there is little doubt that technological advances in decompilers will make the process easier in the future, it is unclear whether advances in encryption will outstrip these advances.[229] In addition, decompiled software is often difficult to understand and to replicate,[230] and again, this difficulty grows with the complexity of the program being decompiled.

The practical prospects for reverse engineering thus remain unclear. Both routes become increasingly cumbersome as the programs that they target increase in complexity. Protection against reverse engineering is thus most likely to be important to developers of relatively simple applications. Developers of sophisticated operating systems may learn that trade secret protection retains a great deal of value despite the technological and legal advantages that may be given to reverse engineers.

The practical aspects of the techniques aside, however, reverse engineering is but one of the issues that have led the courts to oscillate between the underprotection and overprotection of software. The Manifesto viewed the problem as inherent in the nature of software:

> [T]he Office of Technology Assessment once stated that if copyright did not protect more than the literal code of computer programs, it would protect too little, and if it protected more than the literal code, it would protect too much. We agree and make a similar point about patents: Most

---

[224] Sega Enters. Ltd., 977 F.2d at 1527. *See* Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091 (1995) (providing an in-depth discussion of the importance of the court's willingness to recognize this anomaly—and of the dangers posed by courts who rule the other way); Cohen and Lemley, *supra* note 73, at 21-28 (contending further that a right to reverse engineering is so important to the development of a healthy software industry that the courts should find such a right even for patented software. In the absence of a judicial ability to find such a right, they argued that one should be legislatively created).

[225] *See* discussion in *supra* note 219.

[226] *See Sega Enters. Ltd.*, 977 F.2d at 1515 n.2.

[227] *See* Manifesto, *supra* note 11, at 2342. Recall that the Manifesto was published four years before the DMCA was enacted.

[228] *See* Cohen and Lemley, *supra* note 73, at 16-21.

[229] *See* 17 U.S.C. § 1201(g) (2000) (protecting advances in encryption).

[230] *See Sega Enters. Ltd.*, 977 F.2d at 1515 n.2.

> of the commercially significant innovations in software will be underprotected if patent law adheres to its traditional bounds; yet if this law is stretched to protect commercially valuable program innovations, it will overprotect them.[231]

The Manifesto further illustrated this underprotection/overprotection cycle by reference to a few well-known cases.[232] In *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*,[233] the court ruled that everything about a program except its basic functionality was protectable expression, unless there was only one way (or very few ways) to achieve that functionality, in which case the idea and its expression were merged and no protection was available.[234] This decision rendered virtually all aspects of a program protectable, and set the stage for a period of overprotection.

The aforementioned *Altai* filtration test turned the tide in the other direction.[235] Again, *Altai* introduced a more sophisticated approach towards distinguishing between protectable and unprotectable elements of a program.[236] The Manifesto noted that a scrupulous application of the *Altai* test could strip the protection from virtually all non-literal aspects of a program and thus lead back to underprotection.[237] The Manifesto thus contended that most of the courts that claim to have adopted the *Altai* test performed considerably less filtration than the test itself appears to require.[238] It is particularly interesting that the Manifesto cited *Sega* as one of the cases that adhered to the *Altai* viewpoint,[239] despite *Sega*'s willingness to deviate from traditional copyright principles by refusing to protect one of a program's undisputed literal elements (*i.e.*, its source code).

This cycle of overprotection and underprotection, and the seeming anomaly of allowing decompilation despite the general contours of copyright law, highlight the third key difficulty that the Manifesto found with the current regime—the lack of disclosure. Because current IP rights in the absence of trade secret protection (and in the absence of judicially-crafted exceptions to standard copyright principles, such as the one adopted in *Whelan* and rejected in *Altai*) tend to underprotect the behavioral aspects of software, many

---

[231] Manifesto, *supra* note 11, at 2356 (citing Office of Technology Assessment, U.S. Congress, Intellectual Property Rights in an Age of Electronics and Information at 78 (1986)).

[232] *See id.* at 2356-61.

[233] 797 F.2d 1222 (3d. Cir. 1986).

[234] *See id.* at 1248.

[235] *See Altai*, 982 F.2d 693.

[236] *See id.* at 702-9.

[237] *See* Manifesto, *supra* note 11, at 2360-1.

[238] *See* Manifesto, *supra* note 11, at 2361 (citing Dennis S. Karjala, *Recent United States and International Developments in Software Protection* (pt. 2), 16 EUR. INTELL. PROP. REV. 58, 60-62 (1994)).

[239] *See* Manifesto, *supra* note 11, at 2359 n. 201.

developers guard their source code zealously. This guardianship runs counter to the basic *quid pro quo* underlying the IP system.

Thinking back to first principles, authors and inventors were supposed to be rewarded for promoting the arts and the sciences.[240] Sharing new knowledge is an obvious and important part of this promotion. As a result, patent holders are required to disclose their innovations in a description that can be understood by an average practitioner of the relevant discipline.[241] Copyright holders typically reveal their works to the world as a matter of course; any competent reader should be able to discern both the protected expression and the unprotected underlying idea.

Software developers, however, are in a bind. Were they to reveal their source code in a manner that an average programmer could understand, they would be shorn of all meaningful protection. The construction of an independent expression of the underlying idea would be a trivial task. Competitors could thus enter the commercial market with impunity. Original developers would have little to show for their efforts and would thus be under-compensated and inadequately motivated. At the same time, their current ability to withhold source code requires their competitors to undertake a great deal of duplicative work. This reconstruction represents an inefficient use of resources and thus fails to promote adequate progress in the field. The Manifesto recognized that some mechanism was needed to encourage developers to publish their source code without depriving them of the protection necessary to earn a viable return on their R&D investments.[242]

These three key areas of difficulty played a central role in the Manifesto's construction of a proposed alternative regime.[243] The basic elements of their proposal were that:

- Software developers seeking IP protection would have to register their source code. When they were granted their rights, the source code would become publicly available.[244]
- Developers granted IP protection would retain exclusive rights to behavioral innovations embodied in their source code. All competing products that embodied that behavior would be considered infringements.[245]
- The protection would be of relatively short duration. It would be much shorter than either a patent or a copyright in the current regime, and

---

[240] *See* U.S. CONST. art. I, § 8.

[241] *See* CHISUM ON PATENTS, *supra* note 18, at § 7.01 et seq.

[242] *See* Manifesto, *supra* note 11, at 2364-65.

[243] While this terse statement leaves many details unspecified, it does capture the most significant features of the proposal forwarded in the Manifesto (which was rather spare on details itself). Thus, while it might be technically correct to refer to the proposal, as stated, as one variant on a theme launched in the Manifesto, this article will refer to it simply as "the Manifesto's proposal."

[244] *See* Manifesto, *supra* note 11, at 2365-71.

[245] *See id.* at 2371-78.

would expire long before the expected useful lifetime of the innovations that it protected.[246]

This proposal, if adopted, could lead to a restructuring of the software industry. It would certainly realign the rewards available to software developers. It would affect their decisions about the relative merits of IP and trade secret protection. It would thus motivate different types of information exchanges, marketing programs, pricing strategies, and commercial contracts. It would also result in a much larger percentage of available source code being shared.

In Stage 2 terms, the Manifesto's proposal promises broad, short protection that is shallow on the underlying knowledge but deep in commercial uses—in many ways similar to patent law. One of the greatest potential benefits of this type of regime would be its ability to create something akin to a blocking patent. The regime's breadth would guarantee the first mover a valuable monopoly, but only for a limited time. Its shallowness on knowledge would allow competitors to combine the first mover's research with their own, and again, the time limitations would provide them with an opportunity to capitalize on the combined research while it was still valuable.

C. *Incentives and Responses under Alternative Regimes*

Before considering the relative merits of the current regime and the Manifesto's proposal as required by Stages 3 and 4, two related points that the Manifesto failed to discuss in any detail must be raised. First, under any reasonable IP regime, potential applicants may opt out of the IP system and maintain their innovations as trade secrets. Second, as discussed above, software is not a monolithic industry; it includes both platforms and applications. While participants in these sub-industries share many concerns, they also play in different markets. The differences between these markets can have a profound effect on the ways in which revenues are generated and thus on the way that developers seek rewards for their innovations. The differences can lead to very different marketing strategies and to different decisions about the relative merits of IP rights and trade secret protection. An analysis of the software industry must thus consider the likely reactions of both market segments. In particular, it must consider the circumstances under which the members of each segment might consider opting out of the IP system and relying, instead, on trade secret protection.

1. Applications

Dual copyright/trade secret protection is valuable to both platform and application developers, but in subtly different ways. Recall that an application developer's revenue is driven by sales. In the absence of any IP protection, the developer would invest substantial R&D costs, sell a very small number of copies of the software (possibly one) and then be faced with immediate

---

[246] *See id.* at 2378-2405.

competition. Competitors would emerge in two ways. First, consumers who purchased the first few copies could make multiple additional copies at close to zero cost. Because these early purchasers do not need to recoup R&D outlays, they would be able to circulate their copies at close to zero prices and still turn a profit. Second, even assuming that the application developer kept its source code secret, the mere existence of a software program would allow trained engineers to perform at least partial reverse engineering and thus to develop competing software that is effectively a functional equivalent to the original release.

The original developer's revenues under this scenario would thus be restricted to two main sources: the sale price of those first few copies, and aftermarket service and support. Because the service and support markets are inherently porous, the developer will only be able to charge rates that are competitive with service organizations that did not incur R&D costs. It is thus difficult to see any way that the developer could recoup its R&D investment without charging its first few customers an exorbitant price. Given this imperative, most potential customers would prefer to wait until the price dropped near zero. The developer's prospects of recouping its R&D costs are thus slim. A regime in which software application developers receive *no* formal IP rights and must rely solely on trade secret law appears to motivate little investment in innovation.

The right (and ability) to preclude copying and distribution is thus central to the motivation of application developers. While some attempt has been made to achieve this goal through contract law in the form of shrink-wrapped licenses, the viability of these licenses remains unsettled.[247] In addition, a contract that allows a vendor to retain rights in the object being sold might violate the antitrust laws as a vertical contract between parties explicitly designed to restrict output and to raise consumer prices. Some sort of underlying IP right covering copying and distribution is probably necessary to carve a meaningful exception, and thus appears to be needed to motivate the developers of application programs.

---

[247] See DIGITAL DILEMMA, *supra* note 70, at 100 (enumerating several cases involving shrink-wrap licenses, only some of which were upheld). The principles differentiating those that were upheld and those that were disallowed remain somewhat unclear. In addition, the very nature of shrink-wrapped licenses raises some troubling questions under the IP laws. If these licenses are allowed for software, should they be similarly allowed for other copyrighted works? Should a book publisher be permitted to include a license prohibiting fair use of the book? If so, is the Copyright Act reduced to nothing more than a default licensing scheme? These questions notwithstanding, the Uniform Computer Information Transactions Act (UCITA) § 209 (approved in 1999 for recommendation to the States by the National Conference of Commissioners on Uniform State Laws) recommends that all States adopt provisions recognizing the validity of shrink-wrapped (or "mass market") licenses. *See* UCITA § 209, *available at* http://www.ucitaonline.com/ucita.html (last visited Jan 9, 2002). This provision remains controversial (as do various other parts of UCITA).

An immediate leap from this motivational need to standard copyright protection would cut the analysis short. The appropriate questions must now focus on the Stage 4 assessment of the societal costs and benefits inherent in various combinations of depth, breadth and length, and on the Stage 3 consideration of ways in which those combinations impact private sector investment decisions. Once again, the goal of this analysis must be to find suitable rewards for potential investors in R&D without unduly burdening would-be competitors, consumers, or the general growth of knowledge. A consideration of the current regime is again instructive.

The most glaring oddity of the current regime is its length. Copyrights last at least 95 years,[248] a time frame that is effectively infinite in the software world. Under this standard, most UNIVAC code (written in the 1950's) retains decades more of protection, and Windows 95 source code will not enter the public domain until late in the 21st century—long after it will have lost any residual value. This situation stands in stark contrast to conventional copyrighted works. Part of the assumption guiding copyright length is that the few creations that retain value at the end of the copyright term retain *immense* value (*e.g.*, Mickey Mouse)—so much so that they have become icons of popular culture whose use by society at large should not be restricted. In the case of software, any components that retain value at the end of this long life will undoubtedly retain that value because they have been incorporated into future generations protected by later copyrights and thus remain protected in their useful incarnations. The decision about protective length is thus whether the protection should expire during the program's foreseeable useful lifetime or be granted to the initial developer for an effectively infinite time period.

The length decision helps focus thinking about depth and breadth. Infinite protection is obviously more valuable than finite protection. Thus, finite protection must be deeper and/or broader than infinite protection in order to motivate the same investment in R&D. Per the discussion above, *any* meaningful protection must bar copying and distribution; the developer must be allowed to sell single copies of object code. From the consumer's perspective, then, a standard purchase of an application program must allow installation on a single computer (perhaps one at a time, allowing a user to upgrade hardware without repurchasing software or to place copies on multiple computers owned by the same individual and never used simultaneously), a maximum range of uses for either personal or commercial benefit, and no rights to copy, to circulate, or to resell the purchased object code (*i.e.*, to take any action that would lead to more than one copy of the object code being used at any given time).

The tricky issues thus focus on the depth and breadth of rights retained by the developer with respect to its competitors, and on the rights reserved to those competitors either as purchasers of the application or as members of

---

[248] *See* 17 U.S.C. § 302(c) (2000) (describing the copyright term for works made for hire, or corporate works); NIMMER, *supra* note 121, § 9.01, 9.08-12.

society. If the first application released proves to be successful, many competitors will want to enter the market. These would-be competitors all understand that the only way to win market share from a successful incumbent is to introduce products that are *both* similar enough to the original product to attract the users who like it *and* superior in enough ways to make those consumers switch.

These dual requirements frame the considerations of breadth and depth. In this context, breadth determines the range of non-infringing, new products that may be released by competitors. Depth determines the range of activities that the competitors may take in incorporating the successful innovation into their own competing products. Stated another way, the depth of the protection dictates the steps needed to reverse engineer the innovation. The breadth limits the ends for which the reverse-engineered information may be used.

Recall that patents may be parameterized as broad, shallow on knowledge, deep on commercialization, and twenty years long. Conventional (*i.e.*, non-functional) copyrights are both narrower and shallower than patents but last much longer. Software copyrights are different, at least in practice. They allow the original developer to withhold the copyrighted source code from public view, thereby impeding a competitor's reverse engineering tasks. Competitors who devise methods for decompiling object code to recreate the copyrighted source code have thus created an unauthorized copy, thereby infringing the developer's IP rights. Courts that have refused to enforce these rights have conceded that their refusal is less than entirely consistent with the general tenets of copyright law.[249] Furthermore, the underlying difficulty of decompiling software (which may increase as cryptographic data-security algorithms improve), and the necessary imperfection of the resulting "copy," makes the de facto combination of software copyrights and trade secrets broader than conventional copyrights. A ban on decompilation would also make them deeper.

Once again, then, the combined copyright and trade secret protection of the current regime provides software developers with protection that is broader than conventional copyrights, possibly deeper than conventional copyrights, and of effectively infinite length. Empirical evidence suggests that this combination of protections has motivated large investments in R&D, generating increasingly sophisticated software applications. It would be hard to contend that U.S. policy towards the protection of software rights has failed to foster a vibrant software industry. From that perspective, the current regime can certainly be judged a success. What is less clear, however, is whether some alternative protective regime would have been (or would now be) preferable. It is possible that some other regime could have generated even more competition, thereby leading to both superior products and lower prices.

With that in mind, the Manifesto's proposal can also be evaluated. Recall that this proposal offers protection that is broader and shorter than the current

---

[249] *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1992).

regime, and that is shallow on the underlying knowledge (*i.e.*, by requiring disclosure of source code) but deep in its protection against commercial competition. The differences between the likely workings of the Manifesto's proposal and of the current regime, at least in the applications market, may be illustrated with the help of a simple example.

Consider the development of a hypothetical new, popular application, the widgetizer. Under the current regime, several competing software firms would be likely to realize that widgetizing constitutes a huge potential market, and compete to be both the first to market and the best early product to market. Several first-generation products would be released, all produced through independent research efforts. Widgetized files generated on one first-generation product would probably be incompatible with those written using any competing product. Consumers would likely be divided over the features and systems that they liked. The competing widgetizer firms would be able to see which competing systems fared well, and to the extent that these firms could derive comparable functionality independently, they could incorporate their competitors' strengths into their own second-generation systems.

Between the first and second generations, some firms would likely exit while others would enter. Second generation products would incorporate new features—some newly devised, others inspired by competing first generation products. As the generations progressed, increasing numbers of features would be incorporated into each product, and some would even become compatible with old versions of competing programs. Eventually, most competitors would be likely to leave the field, and the competitive tension that motivated early improvements would dwindle.

In a regime designed along the lines of the Manifesto's proposal, the market would evolve differently. Suppose that the first mover had been awarded exclusive rights to market a widgetizer for a brief period in exchange for publishing its underlying source code. The rights holder would have to recoup its entire investment during this monopoly period, because its competitors would undoubtedly continue their research until the rights had expired. Once the competitors gained access to the rights holder's source code, they could develop additional features, for which they, in turn, could seek IP protection. They could then cross-license their newly protected features with the first mover to market a (protected) second-generation program, or they could develop their own program and wait for the monopoly period to end.

If the firms choose to cross license, consumers will gain rapid access to a second-generation widgetizer compatible with their existing programs, albeit at a monopoly price. Competition would then move to the next set of features. If cross licensing did not occur, it is likely that several competitors would have each devised their own sets of (protected) features. The second-generation would thus have consisted of several competing systems, each with a unique set of advanced features and each compatible with the first generation monopoly product. Because the monopolies granted in the second generation would be much narrower than the first generation monopoly, the competing second-generation products would begin to constrain one another's prices.

Third-generation systems, in turn, would incorporate all desirable second-generation features—and be compatible with all second-generation systems. The process would continue from there.

This comparison suggests that over the first few generations, the Manifesto's proposal would almost certainly have increased compatibility among competing systems. At the same time, it would probably also have sped technological development and, on the down side, elevated early prices. The potential increase in compatibility is particularly significant given the network nature of the software industry. By forcing protected software firms to disclose their source code, a *de facto* open standard would have been likely to emerge.[250] This open, public domain standard would present fewer barriers to entry than do the current privately-owned standards of many critical applications—thereby leading to a larger number of firms competing to enter successive technological generations, and possibly (eventually) to lower prices. These considerations—the early price/quality tradeoff, the emergence of different standards, the potentially larger marketplace of vendors, and possibly even the eventual reduction in the prices that consumers pay for high quality goods—all play central roles in the Stage 4 analysis of the relative societal values of the current regime and the Manifesto's proposal. [251]

---

[250] There is a substantial literature on standards, standard setting bodies, and strategies for allowing a technology to emerge as a *de facto* standard in a network industry. For a general introduction to this literature, *see, e.g.*, SHAPIRO & VARIAN, *supra* note 162; Stanley M. Besen and Joseph Farrell, *Choosing How to Compete: Strategies and Tactics in Standardization*, 8 J. ECON. PERSP. 117, 117 (Spring 1994).

[251] The idea that an open, public domain standard would benefit the public is hardly novel. In fact, it is the basis of an entire movement within the hacker community—known first as the "Free Software" movement, currently as the "Open Source" movement, and tracing its roots to the earliest versions of Unix. *See The Open Source Initiative: History of the OSI*, *at* http://www.opensource.org/docs/history.html (last visited Dec. 31, 2001). The movement's "social contract" states (in part):

> Open Source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria: 1. Free Redistribution . . . . The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale . . . . 2. Source Code . . . .The program must include source code, and must allow distribution in source code as well as compiled form . . . . Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed . . . . 9. License Must Not Contaminate Other Software . . . . The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Id. It should be clear that had the software industry developed along the lines advocated by the open source movement, it would look quite different from the way that it does today. Members of the movement, of course, contend that this but-for software industry would have improved the array of products available to businesses and consumers at prices below those assigned to current offerings. For more information on the Open Source movement,

Had the Manifesto's proposal been in place throughout the history of commercial software development, its realignment of rights and incentives would have distributed the rewards of software innovation very differently throughout the private sector. In a network governed by a privately owned standard, the network owner reaps huge rewards. The closer the network comes to an open standard, the closer the industry comes to being fully competitive. Successfully innovative firms would still receive fair, and frequently substantial, returns, but nowhere near the monopoly profits that can be extracted by a network-owning monopolist. This redistribution thus frames the Stage 3 comparison of the private value of the protective regimes.

At this point, is important to recall that the *raison-d'être* of this entire analysis is the determination of *societal* values for each of the regimes—in other words, the outcome of the Stage 4 inquiry. As noted above, the consequences of following the Manifesto's proposal seem likely to include higher quality software, lower prices in the long term, increased competition, and fewer opportunities for anticompetitive behavior, all without sacrificing innovation. These consequences appear to confirm both the Manifesto's claim about the superiority of its proposal and this article's claim about the merits of industry-specific IP rights at least with respect to the part of the software industry focused on applications.

That said, it is also important to recall the caveat of section V.B that an evaluation of a proposal's merits is incomplete without a corresponding consideration of transaction and transition costs, an issue that will be deferred until section VII's summary analysis. In the meantime, it is fair to conclude the consideration of applications software by saying that the Manifesto's proposal appears to be superior to the current regime—at least on its merits.

2. Platforms

The analysis of incentives in the applications market began by considering the hypothetical problem of underprotection, thereby motivating the need for some sort of IP protection. In the platform market, the more pressing problem appears to be overprotection. This difference emerges from the difference in primary revenue sources associated with the two classes of products. Recall that while the primary source of application revenues lies in product sales, the primary source of platform revenues may lie in network access, which is perhaps most easily realized by licensing (or by providing development kits for or pre-release information about updates to) the APIs. The key to assuring this revenue stream lies in keeping access to the platform's source code secret and proprietary. While IP protection could certainly help secure the secret (*i.e.*, by complicating reverse engineering efforts), the best protection is likely to lie in the platform developer's own efforts to assure the security of its crucial trade, including efforts in advanced encryption technology.

---

see The Open Source Initiative, *at* http://www.opensource.org/ (last visited Dec. 31, 2001).

Reduced IP protection would have some potentially significant effects on platform developers. First, it would probably lead to at least a mild increase in their investment in security and encryption. Second, it would almost certainly eliminate their second revenue stream—the sale of copies of their platform—because unprotected platforms could be copied and distributed at close to zero cost.[252] Thus, in the absence of IP rights, platform developers would be likely to increase their returns on service contracts, warranties, support, API access, and development tools and kits, thereby elevating prices in these markets to the detriment of both direct consumers and application developers (who, in turn, would be likely to raise their own prices to consumers). Even these increases are far from guaranteed, however, because the rapid, free dissemination of the platform could help the developer grow its network, thereby yielding increased returns through higher volume sales into the larger aftermarkets rather than through higher prices.

The greater danger in thinking about platforms lies in overprotection. This danger becomes greatest when a single platform developer has been able to achieve monopoly power—a situation that is not only predicted by the economic theories of network markets, but that has actually occurred in the PC world.[253] To see this danger, consider the likely relationships among M, a hypothetical platform developer possessing market power,[254] and four other groups: (i) original equipment manufacturers (OEMs) who manufacture the hardware upon which M's platform runs; (ii) applications developers whose software must run on some platform; (iii) competing platform developers; and (iv) consumers/end users of M's platform and associated hardware and applications. These relationships define the potential profit sources that must be considered in the Stage 3 analysis of private sector values.

Before M achieved market power, its relationship with OEMs was essentially symbiotic. Because most of the OEMs' customers wanted to purchase fully functioning systems, hardware had to be sold with a software platform. OEMs thus had an interest in providing their customers with as many different platforms as possible. M's interest, like that of its competitors in the platform market, lay in making its platform available to as many

---

[252] Note that this phenomenon actually serves one of the objectives of the platform developer—the growth of its own network.

[253] Or at the very least, in the world defined by PCs running Intel processors. *See Microsoft*-Appeal, 253 F.3d 34, 51 (D.C. Cir. 2001); *Microsoft*-Law, 87 F. Supp. 2d 30, 35 (D.D.C. 2000).

[254] While the discussion is stated as a hypothetical, it would be disingenuous to pretend that it is not based on Microsoft. Part of this article's argument is that many of Microsoft's general behavior patterns were reasonably predictable outcomes given the types of legal protection that it was granted. Network economics, copyright protection, and trade secret protection combine to create a powerful and robust monopoly. It should not be surprising that a corporation finding itself in possession of such a monopoly should choose to behave as a rational monopolist. *See generally Microsoft*-Appeal, 253 F.3d 34; *Microsoft*-Law, 87 F. Supp. 2d 30; *Microsoft*-Facts, 65 F. Supp. 2d 1 (D.D.C. 1999).

consumers as possible, regardless of their choice of hardware. In this setting, platform developers and OEMs were in rough power parity. Neither side would have benefited from an exclusive agreement that restricted distribution outside the exclusive. Favoritism given by one platform developer to one OEM would complicate its dealings with other OEMs and vice versa.

The network nature of the platform market suggests that this situation is unstable. Eventually, users would tip the market towards one platform that would become a *de facto* standard.[255] As it happened, a critical mass of consumers began demanding that OEMs provide them with turnkey systems running M's platform. These demands gave M market power; M was suddenly positioned to shift the power equation vis-à-vis the OEMs. M gained the power to insist on either an exclusive contract or no contract at all. OEMs who recognized that a refusal to deal with M would shut them out of much of their own market were thus forced to accept M's terms, thereby magnifying the network effect and increasing M's market power even further.

This situation was entirely contingent on M's ability to control distribution channels. In the absence of IP rights, M would have had no such opportunity; OEMs could have purchased a single copy of M's platform and distributed it freely with their hardware. Many OEMs would have been likely to find themselves primarily selling systems configured with M's platform simply because of consumer demand. They would also, however, have continued to offer some systems with competing platforms—at least until the burden of carrying multiple systems outweighed the revenues generated by offering consumers a choice. M, of course, would not have been shut out of the power equation entirely; M could still have negotiated with OEMs over terms involving pre-release information, support, service contracts, updates, etc.

---

[255] Tipping to a standard was inevitable. The requirement that the adopted standard be proprietary was not. The hardware market, for example, tipped away from Apple's Macintosh and towards IBM's PC only after IBM decided to license its PC architecture, thereby effectively turning the architecture platform into a commodity that was quickly mastered by competitors manufacturing "IBM clones." This decision thus paved the way for the next platform level—namely the operating system—to become the valuable proprietary bottleneck in system design. In many ways, Netscape's decision to publish its source code represented a similar attempt to commoditize the world of web browsers—a decision that turned out to be (at the very least) too late to preserve its viability as an independent company. The parallels between IBM's power in the era preceding the commoditization of its PC architecture and Microsoft's current monopoly power as the proprietor of the dominant software platform are discussed in Timothy F. Bresnahan, *New Modes of Competition: Implications for the Future Structure of the Computer Industry*, *in* COMPETITION, INNOVATION AND THE MICROSOFT MONOPOLY: ANTITRUST IN THE DIGITAL MARKETPLACE 155 (Jeffrey A. Eisenach & Thomas M. Lenard eds., Kluwer 1999), *available at* http://www.stanford.edu/~tbres/research/pff.pdf; Timothy F. Bresnahan, *The Right Remedy* (2001), *at* http://www.stanford.edu/~tbres/Microsoft/The%20Right%20 Remedy.pdf. A discussion of Netscape's strategic decision to publish its source code is presented in MICHAEL A. CUSUMANO & DAVID B. YOFFIE, COMPETING ON INTERNET TIME: LESSONS FROM NETSCAPE AND ITS BATTLE WITH MICROSOFT (1998).

Without the threat of withholding a crucial product, however, the negotiations would have remained on a more equal footing.

This problem suggests the need for a shallower protective regime, at least for platforms. Under the current regime, M may restrict distribution of its platform and place even further restrictions on its resale. A shallower regime could place severe limits on M's ability to achieve either goal. In the shallowest possible regime, M would be granted no rights concerning distribution, and anyone who "found" a copy of M's platform would be free to use, to copy, and/or to resell it.

In a somewhat deeper regime, albeit one that is still shallower than the current regime, M could be granted rights on distribution and on copying, but only until the time of the first sale. In other words, M could be granted strict control of the *number* of copies circulating, but any attempt to restrict activities of the purchaser (either by contract or through IP rights), could be viewed as a misuse of the right and thus grounds for invalidating it.[256] Under this regime, M would receive a fee from the OEMs for every copy that they purchased, but M could not impose any conditions on the way in which the OEMs packaged the platform. Stated in somewhat more conventional copyright terms, this regime would reduce, if not eliminate, M's rights over derivative works.[257]

The Manifesto's proposal takes a different approach. Recall that under the Manifesto's proposal, M's protection in the marketplace would be broad and deep, *but short*.[258] M's registration of its platform, along with the deposit of its source code, would guarantee it a first generation monopoly on certain categories of platforms—namely those that incorporated M's behavioral innovations. The realities of network industries, however, suggest that this short-lived monopoly might be of little value. The true value of a platform monopoly arises only after it has become a *de facto* standard. M designed its platform hoping to earn a long-term platform monopoly. A granted short-term

---

[256] At the time that this article is being sent to press in early January 2002, the ultimate remedy to Microsoft's violations trial remains undetermined, but appears likely to include only behavioral provisions. The analysis presented in this section suggests that some variant of a misuse remedy—under which the courts would refuse to enforce the portions of Microsoft's IP portfolio that relate to Windows until Microsoft could demonstrate that competition had been returned to the market for operating systems on Intel-based PCs—might be appropriate. To date, no one has floated the idea of a misuse remedy vocally and publicly. Most of the debate has centered on the relative merits of structural and behavioral remedies.

[257] A reduction of the rights granted over derivative works has implications to both depth and breadth. From a depth perspective, it expands the range of activities open to the purchaser of an authorized copy, thereby leaving the rights holder with shallower rights. From a breadth perspective, it narrows the scope of works considered similar enough to the protected original to fall within the ambit of its protection. This point illustrates the necessary interaction between depth and breadth—and stresses that not all combinations of depth and breadth are possible.

[258] *See Manifesto*, *supra* note 11, at 2408.

monopoly on a small class of platforms is a weak substitute. As a result, were M forced to choose between the protection afforded by the Manifesto's proposal or trade secret protection, M would probably choose trade secret protection and opt out of the IP system. If M's competitors gained access to its platform's source code, M could never earn its desired monopoly of the platform market. As a result, M might opt for IP rights in a regime like the current one, which uses them to augment trade secret protection. M would be unlikely to opt for any set of rights that forced it to relinquish its trade secrets.

Competing platform developers are M's direct competitors. A shift in IP protection is likely to have a minimal impact on most of these relationships. The only area in which the impact is likely to be felt lies in a slight reduction in the barriers to reverse engineering.

A consideration of the relationships among existing platform developers (or would-be platform developers) suggests why the impact of a regime shift is unlikely to be significant. Perhaps the most interesting questions related to direct competition in the platform market lie in the transition between platforms for successive generations of technology. The theory of network economics suggests that once M has achieved market power for its platform, most of its direct competitors would likely be either relegated to niche markets or forced out of business. Competition is unlikely to reemerge unless and until a new technology arises.

The software industry almost witnessed this type of reemergence with the introduction of Netscape's Navigator and Sun's Java, both of which arose after Microsoft's Windows had consolidated its dominance of the Intel-based PC platform market. These programs have been referred to as "middleware," lying somewhere between true platforms and true applications.[259] These programs promised to combine to serve as a quasi-platform for a new generation of technology—including the technology underlying Intel-based PCs. At one level, they would have interacted with platform APIs, such as the APIs for Microsoft's Windows and/or for Apple's Macintosh systems. At another level, they would have presented open, consistent interfaces to which applications programmers could write new applications. Programmers given this ability would be less insistent on purchasing systems equipped with a specific platform, such as Windows. Middleware thus threatened the ability of the previous generation's victorious platform developer—in the PC world, Microsoft—to exploit the power imbalance created by the success of its platform.

The significance of this middleware challenge to the design of an IP regime is subtle. In the current regime, platform developers possess several ways to generate revenues including both the sale of preferred access to APIs and the sale of copies of the platform. In a regime that did not protect the developer's platform distribution rights, all other revenue sources, including API access fees (frequently in the form of charges for development tools), would become

---

[259] *See Microsoft*-Facts, 65 F. Supp. 2d 1, 9, 19-22 (D.D.C. 1999).

correspondingly more important to platform developers. Middleware would thus become even more of a threat than it is in the current regime. Nevertheless, it is unclear how this difference would affect behavior. The developer in possession of a dominant platform will continue to take all steps possible to defeat middleware threats; potential competitors will continue to try to develop middleware capable of catapulting them to a dominant position in the next generation of the technology.

Thus, competition under the Manifesto's proposal could be expected to continue much as it has under the current regime, although it would be likely to take place among platforms whose developers had opted out of the IP system in favor of trade secret protection. This change could have one of two effects. First, by rebalancing the power equation between platform developers and other private sector players (particularly OEMs), it could reduce the likelihood that any single platform developer could claim victory and push its proprietary platform into the *de facto* standard. This scenario would thus impede the emergence of a standard. Second, by reducing the revenues generated by platform sales, it could force some competing platform developers to exit the market earlier than they otherwise might. This scenario would thus speed the adoption of a *de facto* standard. There does not seem to be any principled basis on which to choose between these scenarios. In either event, there does not appear to be any reason to believe that the adoption of the Manifesto's proposal should affect the ultimate strategic calculus guiding the relationships among competing platform developers.

The developers of application programs are another industry group with whom platform developers must negotiate. This negotiation promises to be more complicated than negotiations between platform developers and OEMs because the technical skills necessary to develop platform and application software are quite similar. As a result, platform developers are also likely to be application developers. Other application developers may thus be developing products that *complement* M's platform but that *compete* with M's applications. M must thus make a careful strategic decision about its relationship with other applications developers.

This decision, which may be described as the tension between *access* and *control*, confronts any firm that controls the keys to a network in a network industry. It determines the degree of exclusivity that M will retain over the virtual network defined by its platform. Recall that one of the defining characteristics of a network industry is that the value of the network grows with the number of network members. Because all application software written for this network must be interoperable with the platform, M can eliminate potential application competitors by refusing to license its APIs and developing all relevant applications software in-house. This approach, however, would motivate all potential competitors to develop applications for competing platforms, thereby complicating M's desire to achieve *and to maintain* its market power in the platform market, as well as depriving consumers of much useful software.

This strategy, which allows M to retain "total control" of its network, can be very risky. In essence, it is a bet by M that its platform is superior to any possible combination of competitor efforts. If M is correct, this strategy could yield enormous profits, although frequently only in the short run. Apple Computers applied a variant of this strategy with consequences that worked to its detriment. In the early-to-mid 1980's, Apple produced hardware/software combinations that were, by most accounts, superior to those of its competitors, the most powerful (and important) of whom was IBM.[260] Apple chose to hold its technology proprietary, thereby making it difficult for competitors to develop Apple-compatible applications.[261] IBM licensed its technology, thereby motivating competitors in a variety of industries, ranging from semiconductors to software, to develop products compatible with its network.[262] The combined efforts of all members of the IBM-originated network eventually surpassed those of Apple, and led to a rapid downturn in Apple's fortunes.[263] This basic story has been repeated in many industries and accentuates the dangers of taking a strictly controlling approach.

The polar opposite of a total control strategy is total access. Under a total access strategy, M would grant all comers access to the network defined by its platform by making its APIs widely available at a low price (or even free of charge). The maximum number of applications developers would thus work on software compatible with M's platform, motivating both developers and consumers to purchase/adopt M's standard. M would thus stand to gain a huge share of the platform market but potentially *no* share of the applications markets. Most commercially viable long-term strategies lie between the extremes of total access and total control.

A restructuring of IP rights is likely to have a profound impact on M's thinking about the relative merits of access and control. An inability to control the copying and circulation of object code must, by necessity, shift M's focus towards maximizing the revenue streams attributable to other revenue sources, such as training, customer service, API licensing, and development tools.[264]

---

[260] *See Apple-History.com* (last visited Dec. 5, 2001), *at* http://www.apple-history.com/history.html.

[261] *See id.*

[262] *See id*.

[263] *See* Leigh Kimmel, *Apple Computer Inc.: A History*, *at* http://www.geocities.com /Athens/ 3682/applehistory.html (last visited Dec. 5, 2001). IBM's architecture is now the basis of the WINTEL standard—named because of the importance of Microsoft's Windows and Intel's microprocessors to that architecture, two products owned by IBM's competitors. Thus, while the architecture won and IBM has hardly fared poorly, IBM both risked and lost leadership of its own architecture.

[264] The "Open Source" movement, *see supra* note 251, which advocates the liberal circulation of source code, sells copies of its source code and thus eschews fees both for the circulation of individual platforms and API fees. The best-known product of this movement currently available is Linux, an open-source version of the Unix operating system. Companies poised to profit from Linux, such as Red Hat and VA Linux, tend to view

Again, if M were forced to choose between disclosing source code and maintaining it as a trade secret, the code would probably remain secret. This decision would relinquish M's rights over distribution of the platform and thus render the total control strategy untenable. M would thus *have to* pursue an access-oriented leveraging strategy as a simple matter of practical economics.

That realization does not end the analysis. Without IP rights, the application competitors with whom M shared the APIs would be under no legal obligation to refrain from publishing and/or circulating them, thereby making moot yet another source of M's revenue. The likelihood of this occurring, however, is rather slim. Application developers will refrain from circulating APIs for two reasons. First, they may be bound by contract to treat the APIs as M's proprietary trade secrets. Second, and more to the point, access to an API, particularly during the pre-release stage, provides an application developer with a tremendous market advantage. An application developer who shares the API with a competitor is squandering that advantage. Coalitions of application developers who agree to present a "united negotiating front" to M would almost certainly be violating the antitrust laws as a monopolistic cartel.[265] Thus, the practical economics of the matter not only constrains M's strategic decisions, but also provides M with *de facto* protection.

On the flip side, M might decide that it was not appropriately leveraging what should be its competitive advantage in the applications markets. Because M has sole access to the platform's trade secret-protected source code, and thus advance knowledge of all pending upgrades, M *should* have a substantial advantage in the development of applications software. M could choose to exploit its position without restricting access by delaying its information sharing. In-house developers could thus benefit from a window of exclusivity in their attempt to develop the best applications, or at least to be the first to market. This window would not guarantee that M would always develop the best applications. It would, however, give M an edge over its competitors, whose applications would have to be either truly novel or appreciably better than M's products to win.

It is not clear how IP rights would have any direct impact on M's ability to pursue this (or other) potentially anticompetitive courses of action because factors unrelated to IP law may constrain M's behavior. It is likely, for example, that a reputation for selling "damaged goods" in the form of slightly stale APIs would reduce the licensing fees that M could generate by selling those APIs. The more central API licensing fees are to M's overall strategy, the more likely M is to ensure that they yield top dollar. While this concern is far from a guarantee that M will deal fairly with its competitors in the applications markets, it does suggest a somewhat higher risk associated with

---

support and service as their main sources of revenue.

[265] *See* PHILLIP E. AREEDA & HERBERT HOVENKAMP, ANTITRUST LAW: AN ANALYSIS OF ANTITRUST PRINCIPLES AND THEIR APPLICATION chs. 7-8 (2d ed. 2000).

anticompetitive behavior and thus a somewhat reduced probability that such behavior will occur.

It appears, then, that the relationship between a platform developer and application developers is essentially governed by the trade secret protection maintained on the platform's source code and pre-released APIs. The major protection that the IP system can afford in this relationship is protection against reverse engineering. A deep IP system that prohibits all forms of reverse engineering would strengthen the platform developer's ability to extract API license fees. A shallow system allowing most or all reverse engineering efforts could weaken the platform developer's negotiating position. It is not clear, however, how much of a limitation that would be. Lead-time is crucial in software. An application developer forced to wait for a new system release before updating her product would be at a clear disadvantage to a competitor who paid for pre-release API access. While this market reality does place some limitation on the platform developer's negotiating strength, the significance of this limitation remains dubious. All told, even major shifts in the IP protection afforded to the platform developer are unlikely to have more than a minor impact on these relationships.

This Stage 3 analysis of the relationship between a platform developer and other private sector actors thus suggests that platform developers forced to choose between trade secret protection and an IP right will opt for trade secret protection. Proposed regimes that force platform developers to make this choice (*e.g.*, the Manifesto's proposal) would probably result in most platform developers opting out of the IP system. Platform development would thus be restricted to a few large, well-funded companies who could afford to circulate enough copies of their platform to develop a network and to reap the bulk of their returns through licensing and aftermarket support fees only after the network had achieved a workable level of popularity.

While this possible shortage of platform competitors is certainly a negative factor that needs to be considered in the Stage 4 analysis of the Manifesto's proposal, it is not clear that the absence of IP rights would chase any existing competitors from the field. The intense need for up-front capital appears to be endemic to the network nature of the platform industry and is thus unlikely to either invite or deter many potential entrants under any type of protective regime. With that thought in mind, the Stage 3 private sector analysis of the platform industry may now feed into the Stage 4 consideration of the likely societal consequences of the proposed reform.

The Stage 4 focus on societal perspectives is probably best understood through the lens of one of society's many personae: consumers. Software consumers are the end users who pay for platforms at prices set under the rules allowed by an IP regime. Under the current system, most platforms are priced as a relatively small component of a total system that combines hardware, a platform, and applications. Were the system reformed along the lines of the Manifesto's proposal, and were platform developers to choose trade secret protection over IP rights, access to a working copy of the platform would become essentially free to consumers. At the same time, consumers could also

expect to pay more for training, support, service, warranty contracts, and possibly applications (whose developers were likely to have incurred higher API access fees, which they are likely to pass through to end users).

The indirect impact of a regime change on consumers is likely to be even more profound. A platform developer's inability to restrict channels of distribution would lead to a radical rebalancing of power between platform developers and OEMs and to a less pronounced rebalancing of power between platform and application developers. These realignments would reduce the ability of a platform developer, including one controlling the *de facto* standard, to extract monopolistic terms and conditions from other vendors in the supply chain. The likely outcomes of this industry restructuring thus include an increase in consumer choice, an increase in competition, a change of the rate at which a *de facto* standard is adopted, a reduction in incentives to platform development, and a consequent possible reduction in platform innovation.

The tradeoffs implicit in these likely outcomes, like those discussed in the context of application developers, frame the Stage 4 assessment of societal value with respect to platform developers. The current regime and the Manifesto proposal once again provide a choice among combinations of likely price, quality, access, and compatibility. If the effective elimination of IP rights for platform developers would reduce neither the number of competitors in the platform market nor the innovative effort that they expend trying to make their platform-defined networks become the *de facto* standard, the Manifesto's proposal would maintain all of society's benefits while reducing societal costs. It would thus appear to be a step in the positive direction. That conditional clause, however, is far from certain. While it is possible that the rewards associated with the ownership of a network standard are so great that no further motivation is needed, it is also possible that neither sales revenues nor network control is enough, in and of itself, to motivate the expensive high-risk challenge of platform development, and that developers forced to make such a choice will instead choose to expend their efforts in other directions. If this contrarian view is true, the Manifesto's proposal could be a disaster.

Such a "disaster," however, need not be crippling. Recall that the initial theoretical discussion of reforms in section 4.3 noted a frequent synergy between radical and conservative reform. Under that general formulation, a radical reform step might be followed by a number of empirically observed negative consequences. Conservative patches would then be needed to address those problems without undermining the basic structural nature of the radical reform. It would be naïve to believe that the first broad-brush pass at a reform as complex as industry-specific IP rights tailored to the software industry could be correct in all of its specifics. Should the Manifesto's proposal be adopted, it must be viewed as the *start* of a new process amenable to corrective tinkering, *not* as the endpoint in a discussion of reform. The Manifesto's proposal would uproot society's value calculations from their current locale and land them in a radically different place. Conservative incremental steps would undoubtedly be needed to move from that new place towards a societal optimum. Foreseeable potential disasters like a reduction in platform innovation suggest

areas in which empirical evidence should be monitored and conservative patches should be considered.

This Stage 4 societal analysis thus highlights yet another significant difference between the application and platform industries. While the Manifesto's proposal appears to help society in its dealings with application developers (and with platform developers who are content to ignore the network nature of their products and treat them as if they were applications), its impact on society's relationship with platform developers focused on network economics remains unclear. On the one hand, it could clear up many of the existing antitrust problems without deterring innovation or retarding progress. On the other hand, it could simply shift the tension from anticompetitive behavior to the inadequacy of IP rights, thereby necessitating a round of conservative tinkering following the radical reform. It is also possible that political transition costs will render any radical reform untenable. Even in this case, the four-stage framework will have served an important purpose. It will have focused the public policy debate on fidelity to first principles—even where the resolution may eschew those principles in favor of expectation interests entrenched in the status quo.

3. Forcing a Choice

The ability of a platform developer to exploit copyright and trade secret law simultaneously increases the prospects of anticompetitive behavior. Unlike application developers, to whom copyright protection is almost certainly the more valuable of the two, many commercial platform developers forced to choose would be likely to retain trade secret protection, to allow their networks to grow organically at zero cost, and to regulate access to other firms in the vertical chain leading to a fully-integrated hardware/platform/application package.[266] Their revenue reductions would come from the stream that is theoretically most expendable in a network industry (*i.e.*, the charge for joining the network), and their loss in strategic positioning would likely preclude more anticompetitive behavior than valid competition. While this loss in distribution revenues and monopoly rents would probably dissuade some investment in platform research, the ample rewards available to anyone who controls a key

[266] Again, the "Open Source" movement, *see supra* note 251, presents a model of platform developers who have opted out of trade secret protection in favor of other sources of revenue. While the movement may yet succeed with Linux, its commercial successes to date have been rare. In fact, the open source movement predates the current dominant PC standard of Windows. While open source advocacy was strong in certain academic circles, however, Microsoft was able to attract large capital investment and to develop the platform that eventually became dominant. It is hard to see how the Microsoft model could have worked in an open source environment. Thus, past experience suggests that while an open source environment could prove profitable, a secretive environment promises even greater profits. Many commercial platform developers are likely to follow this model unless and until its superior profitability potential is disproved.

element of a network's infrastructure is likely to spur a substantial amount of innovation and continued investment.

The Manifesto's proposal, if followed, is likely to bifurcate the software industry. Platform developers will opt out of the IP system in favor of trade secret protection. Application developers will reveal their source code in exchange for the new *sui generis* software rights. As the Stage 4 analyses showed, this bifurcation is likely to improve societal value with respect to the parts of the industry that opt in. The societal impact of the parts of the industry likely to opt out is harder to predict, yet still potentially favorable to the Manifesto's proposal. Considering all factors, then, the analysis appears favorable for industry-tailored rights of the sort proposed in the Manifesto.

VII. ANALYZING THE SOFTWARE INDUSTRY

The preceding section presented a fairly lengthy discussion of the software industry. The discussion included several pointers to relevant conclusions set within the analytic framework of section V. The basic purpose of the framework, however, was to ease policy analysis by presenting these conclusions in a compact, summary form. This section thus presents such summaries for the two policy alternatives being studied: the current IP regime and the Manifesto's proposal.

A. *The Current Regime*

Recall that the analysis proceeds in four stages: (i) industry analysis, (ii) regime specification, (iii) assessment of private value, and (iv) assessment of public value. These stages will be considered in turn:

- Stage 1: Characterize the Industry.

Perhaps the most significant of software's attributes is that, as a technical matter (rather than as a legal matter), it is easy to copy and to distribute at near-zero marginal cost. Within the industry, action shifts quickly as new types of programs become technologically feasible, then popular, then integrated into larger systems. Programs also often become easier to use and more robust as they become more powerful. This constant drive toward higher quality can impel innovators forward even after they have succeeded—unless their progress is impeded by improvidently awarded property rights. These two observations govern the basic shape of the industry, the interrelationships among its participants, the types of incentives available to motivate its potential and actual innovators, and the most likely sources of friction among both incumbents and would-be entrants. The tension between the observations makes software an industry that could either be strengthened or weakened by decisions governing IP policy.

In terms of understanding industry incentives, the broad software industry must be subdivided into platform developers and application developers. There are currently only a few players in the platform market, and entry is constrained by the relatively high barriers common in network industries. The overall applications market is highly competitive, although some key

application markets are dominated by a small number of large players. The key item differentiating these sectors lies in the types of business plans likely to succeed. In either sector, returns on investment are generally measured in months to a few years, although returns on sales of applications may begin more quickly than returns on platforms. Low-priced circulation of platforms to build networks can retard initial returns substantially, but promise exponential growth if the network-building plan succeeds. Thus, the most likely source of revenue to a platform developer comes from the sale of network access, while the most likely source of revenue to an application developer comes from the sale of software. This difference works its way into the incentives likely to motivate innovation.

Three key elements thus dominate the relevant profile of this complex industry:

- The industry may be split into platform and application sectors. Application developers earn most of their revenues from sales of recently developed programs. Platform developers earn most of their revenues from licensing access and support to networks that they grow over time.
- Software needs to be protected to retain any value. The valuable item embodied in software is functional behavior, not literal expression. Patents and trade secrets can protect such behavior; conventional copyrights cannot. Sales receipts are contingent on effective protection.
- A good software package can generate a number of potentially lucrative aftermarkets. The program's developer has a natural competitive advantage in most of those aftermarkets. Software developers can thus profit from their innovations even in the absence of protective rights.

These elements govern the types of business plans that will motivate successful software firms. They should also influence the types of IP rights offered to these firms.

- Stage 2: Define the Protective Regime.

Existing software copyrights are narrow, shallow, and long. They are narrow because behavior is not protected by copyright. They are probably at least somewhat shallower than a standard copyright because of the courts' growing willingness to allow decompilation by commercial competitors as part of the reverse engineering process. Their length, although formally ninety-five years, is effectively infinite because copyright protection lasts far longer than the useful life of computer code. The anomaly of software is that innovators do not have to opt out of trade secret protection to acquire IP rights. The *de facto* combination of copyright and trade secret protection is both broader and deeper than the copyrights provided for standard texts and is similarly of effectively infinite length. The current IP system allows software developers to avail themselves of this strong dual protection.

- Stage 3: Calculate the Potential Return on Private Investment.

Investment and expected return patterns in the software industry, as it currently exists, follow some fairly predictable patterns. Up-front costs are necessary to conceptualize and design software. Labor constitutes the major

cost. Software engineers must be highly educated and trained in the specialized fields of software design and computer programming. Because the competition to be first to market is fierce, substantial up-front investment is often required to field a qualified team quickly. The primary factor limiting returns is likely to be uncertainty of success, rather than discounting; returns are likely to materialize within the first few years, or not at all. Expected returns must thus be large enough to account for that uncertainty. Again, investment in a platform is likely to be riskier and may take longer to show returns than investment in an application. Returns must thus also be correspondingly larger to justify the investment. Developers who are second-to-market with software that they developed independently are free to market their software, but they face a major disadvantage vis-à-vis an entrenched competitor. Eventually, however, a new entrant will be able to dethrone an incumbent who fails to invest in technological advancement.

- Stage 4: Consider the Societal Costs and Benefits.

The current regime has attracted massive capital investment to the software industry. It has also led to the emergence of one or a few key players with market power in virtually every platform market and in many application markets, and generated an increasing amount of behavior that is coming under antitrust scrutiny. As a general rule, software prices have declined and quality has improved. As an absolute matter, the software industry that has emerged under the existing IP regime must be judged a success. As a comparative matter, however, it is not clear that alternative regimes could not have led to faster, better, cheaper systems.

The transaction costs of the current regime are substantial, but relatively well known. They include the management of the groups at the PTO and at the Copyright Office currently engaged in software issues and the litigation and court expenses associated with software litigation. Transition costs for an incumbent system are always defined as zero. Conservative reforms to the current regime could introduce some transition costs, albeit probably relatively minor ones.

B. *The Manifesto Proposal*

The analysis of a regime based on the Manifesto's proposal proceeds through the same four stages:

- Stage 1: Characterize the Industry.

Many of the basic contours of the industry, as described in the Stage 1 analysis of the current regime, are inherent to the nature of software and thus not dependent on IP rights. Under the Manifesto proposal, industry participants would be forced to choose between IP rights and trade secret protection.[267] It is likely that most platform developers will choose the secrecy route, while most application developers will opt for the newly configured IP rights. These decisions will affect market decisions and market configuration

---

[267] *See Manifesto*, *supra* note 11, at 2342-47.

and will likely lead to two industries that are even more distinct than they are under the current regime. The investment and personnel requirements are unlikely to change under any IP reform proposal.

- Stage 2: Define the Protective Regime.

The proposed IP rights are relatively short (say, a few years). They are both broad and deep with respect to commercial competitors, but retain no depth with respect to the state of scientific knowledge; full disclosure of source code is the *quid pro quo* for IP protection. This requirement forces software developers to choose between trade secret protection and IP rights. Secrets, as always, remain proprietary as long as they are secret but have neither depth nor breadth.

- Stage 3: Calculate the Potential Return on Private Investment.

Again, capital and labor requirements are unlikely to change from the current regime. Developers of fundamentally new applications who succeed in securing IP rights will have to recoup their investment during the abbreviated length of those rights. They will thus have to balance their interest in charging low prices to persuade users to adopt their application against their desire to charge high prices during their brief monopoly period. The combination of disclosure and the narrow temporal window should allow competitors to capitalize on diversion, although they will undoubtedly have to wait longer before seeing any return. Cross licensing of innovations in future software generations may help to spread some of the wealth. All told, the emphasis on diversion in this IP regime is likely to reduce the investment's uncertainty, but increase the time before second-comers can expect to see a return. Discounting is thus likely to play a more important role in this regime than in the current one.

Platforms, maintained by assumption as trade secrets, will be unable to attract any revenue through distribution. Only developers of successful platforms will see more than *de minimis* returns from the various aftermarkets. Those returns are likely to be smaller than they are in the current regime but still substantial. Investment in a platform under this regime is thus both riskier and less lucrative than it is under the current regime. Platform development would thus be restricted to a few large, well-funded companies who could afford to circulate enough copies of their platforms to develop a network and to reap the bulk of their returns through licensing and aftermarket support fees only after the network had achieved a workable level of popularity.

- Stage 4: Consider the Societal Costs and Benefits.

The societal impact of the Manifesto's proposal would be felt in two ways. From the perspective of scientific knowledge, the requirement that software developers publish their source code will almost certainly increase both the amount and the quality of public domain code. It would also lead to increased incidences of open standards and a fundamentally different market structure. Some innovations may be slower to market, but they are more likely to be compatible with existing technology when they do arrive. The likely impact of the proposal would be to create more firms competing within an emergent *de facto* standard and fewer competing standards. All told, compatibility and

product quality are likely to improve, but costs may be higher. At the same time, increased competition and a reduction in anticompetitive behavior may drive margins downward, thereby forcing producers rather than consumers to absorb much of the cost increases. The net change in prices paid by consumers is thus hard to gauge. From the perspective of consumers, these higher quality but potentially more expensive applications may be at worst a mixed blessing. It is not possible to know whether the net effect will increase or decrease quality-adjusted prices.

Platform developers, who are likely to opt out of the IP system in favor of trade secret protection, will add nothing to public domain scientific knowledge. Without that protection, their revenue stream currently generated by software sales would disappear. This loss would have to be balanced by price increases elsewhere, likely in training, support, service, warranty contracts, and possibly in applications (as a pass through of the increased API fees that platform developers are expected to charge application programmers). Again, the net effect of these increases is hard to project *ex ante*.

While the introduction of a *sui generis* regime protecting software may lead to increased lobbying and consequently higher societal monitoring costs, the operational transaction costs of the Manifesto's proposal need not differ greatly from those associated with the current regime. They would continue to include the management of the relevant groups at the PTO and at the Copyright Office or at some new Software Rights Office. It is again difficult to gauge whether, in the long run, the new configuration of these offices would be more or less expensive than the current setup. Litigation will remain inevitable, and would almost certainly increase in the years immediately following a radical policy shift, but if the system is an improvement on the merits, litigation should decrease in the long run.

The "long run" hedges in these transaction costs allude to the likelihood of substantial transition costs. Government agencies, courts, attorneys, and companies will all need to be retrained about the new regime. Until the contours of the regime have equilibrated, litigation and disruption are likely occurrences. Furthermore, litigants whose rights were reduced by the regime change may name the government in a series of takings lawsuits. These transition costs could prove to be so substantial that they could negate the benefits that would otherwise be expected from the regime change. That scenario, however, appears to be unlikely. In the long run, sound industrial policies will generally benefit society.

C. *Policy Implications*

These four stage analyses outline the likely impact of moving from the current regime to one along the lines outlined in the Manifesto. As noted above, most but not all of the anticipated effects seem to favor the Manifesto's proposal. This conclusion is consonant with the article's thesis that industry-specific IP rights can come closer to the societal optimum than can generic rights drawn from the existing one-size-fits-all approach. The article has thus

reached several policy prescriptions—some matters of general methodology, and some specific to the software industry—as promised in section I:

- Proposed reforms, particularly radical reforms, should be evaluated in terms of their fidelity to first principles. The first principles of the IP system exploit property rights to promote innovation. Reform proposals should be evaluated within a framework that highlights the projected costs of the rights offered and the expected benefits of the anticipated innovations. Proposals deemed likely to make a net contribution to society should be adopted. All others should be rejected.
- Industry specifics should be studied in the construction of proposed reforms of IP rights. Technology, timing, resource requirements, and incentives should dictate the types of rights offered to the members of a given industry.
- The value of trade secrets should never be forgotten. Under a variety of circumstances, firms forced to disclose their erstwhile secrets to obtain legal protection may instead choose to keep their knowledge secret. Never ignore the possibility of widespread opt outs when assessing the likely impact of a reform proposal.
- Society can help itself by offering *sui generis* software rights. These rights should protect the innovative behavior captured by the programs, expire relatively quickly, and force developers to disclose their source code.

This section provided a worked example of the analytic framework, demonstrated the likely superiority of an existing reform proposed for the software industry, and paved the way for informed debate about other industries in which *sui generis* protection may be appropriate.

VIII. CONCLUSIONS

The two protective regimes evaluated in the context of the software industry offer different tradeoffs to consumers and to society. They are likely to attract different balances of investment, and to distribute rewards differently throughout the private sector. While the Manifesto's reform proposal was motivated by a desire to avoid many of the pitfalls of the current regime—and in particular the wave of anticompetitive behavior—it will not avoid them without incurring countervailing societal costs. Nevertheless, it does appear to represent a net societal gain—and likely a very large net gain. While the potentially large transition costs caution against rushing to adopt its prescriptions, careful, intelligent steps should be taken in its direction.

The point of this article, however, was more than the demonstration that a specific reform proposal is likely to be superior to existing law. The analytic exercise was designed to show that many of the problems currently plaguing the software industry were inherent in the decision made early in the legal consideration of software that every computer program must fit within one of the few existing categories of IP rights. The first principles approach showed that the investment, incentive, and technological properties of an industry are

crucial to understanding how that industry will interact with a set of property rights. Under this approach, IP rights may be crafted to motivate desired industrial development and to retard (if not to disable) undesirable and anticompetitive behavior. While there may be circumstances under which society would be well served by fitting a square peg into a round hole, such instances are largely fortuitous. Relying on them makes for poor policy. A first principles approach is much more likely to redound to the long-term benefit of society.

In closing, then, it is worth reiterating a few points made toward the beginning of this article. The bifurcation of innovators into authors and inventors may have run its course. The division of IP rights into patents and copyrights—a division that served the country reasonably well during the agrarian and industrial ages—may be insufficient to deal with the complexities of the information age. Advances in the basic natural sciences of biology, chemistry, and physics, and in the basic social science of information, are powering a wave of new industries. The speed of these advances, the structures of these industries, the nature of their products, the necessary interaction among competitors, and the opportunities for profit, may confound attempts to adapt patent and/or copyright law. Software may be among the most mature of these industries, but it is unlikely to be the only one. The next few decades are likely to feature a growing number of such industries and should thus also feature an analogous growth in new formats of IP rights. The lessons learned from the existing formats will prove invaluable. Guidance in the design of all such rights should come from the first principles articulated in the Constitution: harness the profit motive to promote innovation.