COMMENTARY: BLACK HOLES OF INNOVATION IN THE SOFTWARE ARTS

By Mark A. Haynes †

I. INTRODUCTION

In order to promote technological progress, the intellectual property system relies on a careful balance between free access to information about products on the market on the one hand, and the encouragement of investment in innovation through the exclusive rights of patent, copyright, and trade secret laws on the other hand. In the software arts, including e-commerce, the system is unbalanced by the tendency of copyright law to isolate technology in pockets controlled by the original author, and to thus act as the gravity for "black holes" of innovation.

The Magaziner Report recommended that strong patents in the area of e-commerce would further encourage technological development,[1] and some steps have been taken to that affect. The courts have recognized the patentability of software-based inventions,[2] and the software elite have begun to participate in the patent system.[3] As a result, the volume of patent applications being filed in the areas relating to e-commerce is exploding.

However, simply increasing the number of patents in software technologies does not ensure that the goal of promoting progress in the technology is achieved. Copyright laws need to be reevaluated and modified to

---

† B.S.E.E., 1977, University of Texas at Austin.; J.D., 1981, Stanford Law School. Admitted to practice before the U.S. Patent and Trademark Office, 1982. The author is the founding partner of Haynes & Beffel LLP, a patent prosecution firm in Half Moon Bay, California. The author would like to thank the editorial staff of the *Berkeley Technology Law Journal* for their assistance.

1. *See* WILLIAM J. CLINTON & ALBERT GORE, JR., A FRAMEWORK FOR GLOBAL ELECTRONIC COMMERCE § 3 (1997), *available at* <http://www.iitf.nist.gov/eleccomm/ecomm.htm>.

2. *See, e.g.*, Diamond v. Dierh, 450 U.S. 175 (1981); *In re* Meyer, 688 F.2d 789 (C.C.P.A. 1982); Arrhythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053 (1992).

3. *See, e.g.*, Rodney Ho, *Patents Hit Record in '98 as Tech Firms Rush to Protect Intellectual Property*, WALL ST. J., Jan. 15, 1999, at A2; Jonathan M. Moses, *When Copyright Law Disappoints, Software Firms Find Alternatives*, WALL ST. J., May 4, 1993, at B6.

create an appropriate balance in intellectual property laws to promote technological progress in the software arts. In particular, I believe that rules enabling reverse engineering of software-based products should be revisited and amended to allow competing companies and individuals to study each other's work in detail and use the ideas discovered without fear of copyright liability. The reverse engineering right in the Semiconductor Chip Protection Act of 1984[4] may provide a suitable model for legislation in this area.

This paper comments on the role copyright plays, independently and in concert with patents, in blocking innovation in the software arts. Part II describes how copyright slows the pace of innovation in the software arts, particularly by removing much unpatented but important technology from the public domain and preventing effective reverse engineering. Part III discusses how the patent system fuels innovation, and argues that copyright upsets the balance in intellectual property. The paper concludes with a call for further study on the effect of copyright on innovation in the software industry, and legislative reform to further enable reverse engineering.

II. HOW COPYRIGHT SLOWS THE PACE OF INNOVATION IN THE SOFTWARE ARTS

Many comparisons have been made between the programming art and the arts traditionally covered by copyright, such as writing and painting.[5] These comparisons feed the argument that intellectual property law should focus on the protection of developers of code from others who might pick-

---

4. The Semiconductor Chip Protection Act of 1984, Pub. L. No. 98-260, 98 Stat. 3335, 3347 (codified as amended at 17 U.S.C. §§ 901-14). The reverse engineering provision of the Act provides that it is not an infringement of the exclusive rights of the owner of a mask work for:

> (1) a person to reproduce the mask work solely for the purpose of teaching, analyzing, or evaluating the concepts or techniques embodied in the mask work or the circuitry, logic flow, or organization of components used in the mask work; or
>
> (2) a person who performs the analysis or evaluation described in paragraph (1) to incorporate the results of such conduct in an original mask work which is made to be distributed.

17 U.S.C. § 906(a) (1994).

5. *See, e.g.*, Anthony L. Clapes, *Confessions of an Amicus Curiae: Technophobia, Law and the Creativity in the Digital Arts*, 19 DAYTON L. REV. 903 (1994); Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977 (1993).

up and improve their work, as if software developers were artists working in isolation.[6] But it is just this isolation of software technology, induced by copyright, that damages the processes of innovation. The isolation of the developer is important for the content, as opposed to the technology, of e-commerce. Unlike the content on the web, the technology for delivering content needs to be shared and understood by many in order to advance. Copyright laws must be understood to determine how they fuel, or block, the free flow of information.

My experience as a patent attorney, coupled with anecdotal examples from legal scholarship and the market, leads me to believe that software engineers are constantly reinventing the wheel (and patenting old ideas) because key avenues for learning what others have already done are not open to them. Notwithstanding the creativity of individuals in the field, as long as they are blocked from careful study of the work of others much of the creativity is wasted traversing ground exhaustively explored by others before them.[7] Innovation is slowed because creative energies are applied to ignore the work of others and to re-do basic designs in an effort to avoid accusations of copyright infringement, rather than to build on and improve the state of the art, which necessarily includes the work of others.

This Part discusses how copyright effectively blocks important "conduits" that facilitate the flow of information into the marketplace. We can see the effects of this restriction in several (admittedly anecdotal) real-world examples. In particular, the open source movement can be understood in this context as a way to innovate around the stifling effects of copyright on innovation.

A. Copyright Restricts the Flow of Information

Innovation is fed by three key conduits for the flow of information: (1) reverse engineering of products, (2) patents, and (3) technical literature. However, copyright blocks each of these conduits of information. In doing so, copyright essentially provides the gravity for black holes in which no one but the copyright holder is able to innovate, effectively blocking competition for compatible products, and isolating the technology from improvement by anyone but the copyright holder.

First, copyright essentially takes away the freedom to reverse engineer. In fact, with only very limited exceptions, it is copyright infringement to

---

6. *See id.*

7. *See* Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 DAYTON L. REV. 975 (1994).

copy someone else's code to study it, and to apply the knowledge gained to make competing products, under the so-called "intermediate copying" rules.[8] One defense to copyright infringement is "fair use" under Section 107 of the Copyright Act.[9] However, as discussed in more detail below, it offers a poor refuge to those seeking to reverse engineer software to derive the functional aspects. Because fair use is an affirmative defense, the burden is on the person reverse engineering the product to prove fair use, and this alone serves as a deterrent. The defense is further weakened by the presumption that commercial uses are unfair.[10]

Additionally, while two Court of Appeals decisions recognized a limited right of fair use, they were decided on very narrow grounds, and therefore are not much help. In *Atari Games Corp., v. Nintendo of America*, the Federal Circuit recognized that copying and reverse engineering are not copyright infringement when the copying and reverse engineering are necessary to learn the unprotected ideas and processes.[11] The Ninth Circuit decision in *Sega v. Accolade* similarly found that copying and reverse engineering computer programs can be a fair use under the Copyright Act. [12] However, the holding in *Sega* only applies in the narrow circumstance of when a programmer needs to reverse engineer in order to ensure compatibility with his own product, and no other means of access exists.[13] Although both of these decisions potentially allow a fair use defense for reverse engineering, practical realities make this extremely difficult; the ad hoc nature of copyright enforcement, the variability of outcomes, and the need to rely on an affirmative defense makes "fair use" a thin basis for competitors interested in learning the unprotected ideas and processes from computer programs through copying and reverse engineering. As a result, lawyers tell programmers to avoid looking at the work of others for fear of copyright infringement.

---

8. *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); Atari Games Corp. v. Nintendo of Am., 975 F.2d 832 (Fed. Cir. 1992). *See also* John G. Mills, *Possible Defenses to Complaints for Copyright Infringement and Reverse Engineering of Computer Software: Implications for Antitrust and I.P. Law,* 80 J. PAT. & TRADEMARK OFF. SOC'Y, 101 (1998).

9. 17 U.S.C. § 107 (1994).

10. *See Sega Enters.*, 977 F.2d at 1522.

11. *See Atari*, 975 F.2d at 844. Note that the Federal Circuit found that Atari had exceeded the scope of that right. *See id.*

12. *See Sega Enters.*, 977 F.2d at 1520-21.

13. *See id.* at 1521. Unlike in the Atari case, the Sega court found Accolade's intermediate copying to be a fair use. *See id.*

Second, copyright reduces the incentive for software companies to patent their products. The exclusivity provided by patents is expensive, requires a degree of value added to the state of the art, and can be neutralized by improvement patents. In contrast, exclusive copyrights come at virtually no cost to the holder. With fewer patents, companies keep more information about important ideas secret and avoid gauging their work against that of others. So far, patents describe only a small part of the software arts. But many software companies are now quite aggressively applying for patents.[14]

Finally, without free reverse engineering, we are left with the technical literature as a means for sharing ideas in the software arts. But technical publications are generally theoretical in nature, and very shallow compared to real products. Without reverse engineering, there are huge gaps in usable information.

B. Evidence of the Slow Pace of Innovation

To verify the harm copyright has done to the software field by restricting these conduits of information, what we need is a comparative review of the processes of innovation in the software field with other high technology fields not protected by copyright. The goal of this review would be two-fold: (1) to provide insights needed to sort out the intellectual property laws related to computer software, and (2) to restore a balance that promotes innovation and fair competition in the software field, like the balance achieved in other fields of high technology.

In the absence of such a study, anecdotal evidence must suffice. Commentary from the legal community on intellectual property laws as applied to computer software includes some interesting perspectives on the lack of invention in the software field. For example, it has been argued that "innovation in the software field is primarily incremental. Invention in the formal sense of the term is rare."[15] To a patent attorney, this is an oxymoron. Innovation *is* invention. Practically all inventions in all fields are incremental. Inventions are *patentable* only if they have not been made before by another.

Yet another commentator has argued that constant reinvention in the software field suggests that patents are not needed.[16] After all, if these in-

14. *See* Ho, *supra* note 3; Moses, *supra* note 3.

15. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2376 (1994).

16. *See* John Swinson, *Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection*, 5 HARV. J. L. & TECH. 145, 157 (1991).

novations can be so readily remade, there is no reason to encourage dissemination of them, or so the argument goes. This, of course, is nonsense. Re-making of inventions wastes resources, and slows down the art. In the software field, the programmers are constantly going over old ground, carefully avoiding access to the code of their competitors, and re-inventing. Access to the work of others prevents this wasteful effort.

Comments like these may actually be a reflection of the very slow pace of patentable innovation caused by copyright in the software field. If an inventor must start from scratch every time a new product is developed, unable to study and learn from the work of others, it is very likely the inventor will not be using the optimal approach, since he cannot learn from others what is optimal. This taps intellectual resources, and slows down development of the art.

The most obvious evidence of slow moving technology attributable to copyright is Microsoft's Windows operating system. The copyright on Microsoft Windows blocks the flow of information needed for innovation in the field of compatible personal computer operating systems by would-be competitors. In addition, as the operating system grows through creation of derivative works, the copyright expands. Microsoft is able to exploit the imbalance in the intellectual property laws to expand its copyrighted franchise, and to build a monopoly that naturally expands with the copyright.[17]

The brake on innovation that copyright engenders can be seen by simply comparing the rate of new product releases for Windows to the rate for x86 microprocessors. When Windows 95 came out, it was considered a technological non-event. When Windows 98 was released, the primary innovation in this version that people were talking about was the integration of the Internet browser with the operating system. Commercial x86 microprocessors made by Intel and others, on the other hand, have gone through many revisions since 1995. Intel has enough competition in compatible microprocessors from AMD, National Semiconductor, and others to drive innovation.[18] This competition is able to flourish in part because there are no fundamental copyrights blocking access to the microprocessor

17. *See* Mark A. Lemley, *Antitrust and the Internet Standardization Problem*, 28 CONN. L. REV. 1031, 1053-54 (1996). *See also* Robert H. Lande & Strugis M. Sobin, *Reverse Engineering of Computer Software and Antitrust Law*, 9 HARV. J. L. & TECH. 237 (1996).

18. *See* Luc Hatlestad, *Finally Some Competition, AMD and Others are Making a Legitimate Run at Intel*, THE RED HERRING, Apr. 1998, *available at* <http://www.redherring.com/mag/issue53/competition.html>.

art. Generally, innovators in the software arts are working on products other than the most widely used operating system. This slows down the advancement of the technology to a snail's pace.

Other evidence of the slow pace of innovation in the software field is found in the attacks made on the patents that are issuing for software-based inventions. The software industry loudly complains that software patents are being granted for ideas that are old. But why would a software developer believe that his or her idea was patentable and invest thousands of dollars to get a patent if he or she thought it was an old idea? The only answer to this question that is not cynical is that the work of others is not available to the inventors for study.

The "open source" movement in the software field, viewed in this light, is a clear reaction to the closed nature of copyright protected software.[19] According to this movement, software should be distributed with a liberal copyright waiver, and source code made available to anyone that desires to modify or improve it. With open source, products evolve with input from many developers who all benefit from the work of earlier contributors. Evolution leads to efficient, stable, and versatile products. This movement has gained momentum lately with such acclaimed products as the LINUX operating system, the Apache web server software, the PERL scripting language, and a popular e-mail server product called Sendmail.

However, even the open source movement exploits and is hindered by copyright. The most popular open source code products (LINUX and others) are distributed under the General Public License of the Free Software Foundation. The copyright license limits use of the open source code to products that are also distributed as open source, and for which no patent licenses are required. This aspect of open source distribution agreements has been described as "viral."[20] Thus, users of open source risk losing their own copyrights and contract the patent system out of the product. They lose the benefits of the intellectual property system designed to protect their investment.

III. THE PATENT SYSTEM CONTRASTED

To my understanding, the software industry is the only one that requires such a movement to enable an understanding of the work of others

19. *See* Tim O'Reilly, *The Open Source Revolution*, RELEASE 1.0, Nov. 1998.
20. *Id.*

to promote the advancement of technology. The patent system, by contrast, fuels innovation in fields that lack exposure to copyright.

Unlike copyright, the patent system encourages improvement patents, through which competitors are able to neutralize the patent portfolios of others. People who contribute to the advancement of the technology build relevant patent portfolios to trade with one another. Innovators therefore have freedom of action to utilize the best available technology for their products, whether patented by them or by others. In this way, patents ensure that competitors do the work to advance the technology because only contributors obtain relevant, tradable patents. Those that take a free ride do not obtain patents to trade and become exposed to the patents of others to a much greater degree. Free riders usually end up paying greater license fees or being pushed out of the market.

Because patents are based on claims that define the protected technology reasonably well, competitors can often avoid infringement. The process of avoiding patents, more often than not, results in improvements and new thinking, and thereby accelerates the growth of the technology. Compare Intel in the patent-dominated microprocessor art and Microsoft's operating system copyright. Intel introduces improved x86 processors at frequent intervals. Microsoft made minor upgrades to Windows in 1995 and 1998, but Microsoft does not seem to be in any hurry to introduce any new operating systems. Intel is driven to innovate by the patent system, reverse engineering, and the lack of core copyrights that enable innovative competitors like AMD and National Semiconductor to make and sell compatible products. The slower Intel innovates, the more market share it loses to the companies nipping at its heels. In contrast, the copyright on Windows has stopped all would-be competitors cold. In the trenches, no one is working hard to improve the core of Windows—at least not the way Intel, AMD, and National Semiconductor work hard to improve the microprocessor. Microsoft, however, has not suffered the stress of competition needed for real innovation at the core of the operating system. This can be explained in large part by the imbalance in incentives created by copyright.

Copyright tends to protect the accumulation of simple design choices made by programmers, leaving only the fundamental ideas represented by their work in the public domain. Patents, on the other hand, tend to protect ideas that inventors think are fundamental, leaving everything else in the public domain. This contrasting approach to protection has led some to fear patents in the software field. But the evidence in other fields does not bear out this fear of patents. One learns in practice that the patent system is not so much about protecting what is patented as it is about leaving the

unpatented in the public domain. Copyright removes much unpatented but important technology from the public domain, and upsets the balance of the intellectual property system.

IV. CONCLUSION

Microsoft's monopoly, and other smaller, technical black holes fueled by copyright, like the Apple Macintosh operating system or the WordPerfect word processor, are bad for innovation. We should limit the scope of the copyright in software to enable reverse engineering not only for access to the information in the software, but for use of the information to make compatible products.

Perhaps the approach taken in the Semiconductor Chip Protection Act of 1984 could work in the case of software arts. In that Act, normal infringement was to be found based on substantial "similarity," like the copyright test.[21] However, if the accused could prove that legitimate reverse engineering was conducted, the infringement would be found using a stricter substantial "identity" test.[22] If we made such changes, the industry might have the information necessary for advancement, and the freedom to use it. Someone would find a way to push Microsoft to innovate in its core technology, like AMD and National Semiconductor push Intel. If consumers could buy a Windows 95 compatible alternative from someone else, Microsoft's Windows 98 would surely be a real improvement. Microsoft would be forced to invent in order to maintain market share. The software industry would benefit, and the constitutional mandate to "promote the Progress of ... useful Arts" would be served.[23]

---

21. *See* Brooktree Corp. v. Advanced Micro Devices, Inc., 977 F.2d 1555, 1564 (9th Cir. 1992).

22. *See* Brooktree Corp. v. Advanced Micro Devices, Inc., 705 F. Supp. 491, 495 (S.D. Ca. 1988).

23. U.S. CONST. art. I, § 8, cl. 8.