Some Economic Considerations in the Intellectual Property Protection of Software

Kenneth W. Dam
Kenneth.Dam@chicagounbound.edu

CHICAGO

Law & Economics Working Paper No. 26
(2d series)

SOME ECONOMIC CONSIDERATIONS IN THE INTELLECTUAL PROPERTY PROTECTION OF SOFTWARE

Kenneth W.. Dam

THE LAW SCHOOL
THE UNIVERSITY OF CHICAGO

Some Economic Considerations in the Intellectual Property Protection of Software

Kenneth W. Dam[*]

It is hard to think of a more "useful Art" than software.[1] Tens of millions of otherwise inert boxes sitting atop desks throughout offices, factories and homes spring to useful life through software. Software also makes possible a rapidly increasing set of familiar procedures in business and daily life, from payrolls to airline reservations to video games. Many of the machines we depend on in daily life—whether in home, office or auto—are run through software.[2]

Since software is so useful, it may seem remarkable how much controversy has surrounded the development of the intellectual property regimes to protect it. After hesitation, software has come to receive protection by both copyright and, to a certain extent, by patent.[3] After

[*]Max Pam Professor of American and Foreign Law, University of Chicago Law School. I should like to thank, for comments on earlier versions of this paper, Douglas Baird, Dennis Carlton, David Currie, Frank Easterbrook, William Landes, Lawrence Lessig, Malla Pollack, and Richard Posner, as well as to express my appreciation for support from the Law and Economics Program and The Jerome S. Weiss Faculty Research Fund at the University of Chicago Law School.

[1] See Constitution Art. I, Sec. 8, cl. 8, creating the constitutional basis for protection of the "useful Arts" through patent and copyright protection." See discussion of the Constitutional term "useful Arts," infra notes 15-22 and accompanying text.

[2] Although we think of software in connection with computers, software runs many kinds of machines (or perhaps one should say that computers no longer are always recognizable as computers but rather are commonly built directly into other machines). Among the kinds of machines containing software are "microwave ovens, ct scanners, space shuttles, automobile engines, and anti-lock brake systems." Improved Patents for Software Urged at Second Round of Hearings, 47 BNA Patent, Trademark & Copyright J. 357 (1994). Even though some of this software is encapsulated in hardware in the form of microcode, it remains in essence software. As such it receives the same copyright protection as software fixed in more conventional media. nec Corp. v. Intel Corp., 10 U.S.P.Q. 1177 (N.D. Cal. 1989); Allen-Myland, Inc. v. Int'l Business Machines, 746 F. Supp. 520 (E.D. Pa. 1990), 770 Supp. 1004 (E.D. Pa. 1991).

[3] A number of computer software cases involve trade secret protection. This article, however, will focus on copyright and patent. In addition, it will be concerned almost exclusively with U.S. law. Intellectual property protection of software in other countries has taken a variety of paths, although copyright has become the method of choice and copyright protection is now enshrined internationally in the so-called trips agreement. Agreement on Trade-Related Aspects of Intellectual Property Rights, 33 I.L.M. 81 (1994). Software protection is still grossly undeveloped in many countries. Moreover, even

some false starts, a National Commission (contu) recommended copyright protection,[4] and Congress in effect accepted the recommendation in the 1980 amendments to the copyright statute.[5] Subsequent case law was required, however, to establish definitively that patents could also serve to protect software-related inventions.[6]

I. The Controversy over Software Protection

Controversy over intellectual property protection has taken many forms. Some of the specific issues will be explored in detail below, but for now it is useful to identify a few interacting levels of controversy. The first level is strictly legal. Since software, like some earlier technological innovations with which we have since become doctrinally comfortable, does not immediately fit into our preconceived legal categories, much ink has been spilled, for example, over how to assure copyright protection without allowing copyrights on ideas and the extent to which patents can protect software as such as opposed to solely the manner in which software operates on some physical apparatus.[7]

The second level of controversy applies more specifically to copyright. A strongly held view has been that copyright is somehow inappropriate

---

where the law on the books promises protection, piracy remains rampant. See Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 261 n. 13 (5th Cir. 1988). The Business Software Alliance estimated worldwide losses due to software piracy at $12.8billion in 1994, of which $2.2billion was attributable to the United States. The “piracy rate” (non-“legal” software as a percentage of all software on the market) ranged as high as 80 percent in Japan and 99 percent in Indonesia and above 90percent in a wide variety of third world countries, according to these estimates. Business Software Alliance, Press release (April 27, 1994).

[4] Final Report of the National Commission on New Technological Uses of Copyrighted Works (1979) (hereafter “contu”).

[5] contu was created in 1974 as part of the process leading to the Copyright Act of 1976. This 1976 amendment was not explicit on the subject, but the House Report stated that “literary works” included “computer data bases, and computer programs to the extent that they incorporate authorship in the programmer’s expression of original ideas, as distinguished from the ideas themselves.” H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. 54 (1976). contu reported in 1978 and Congress in effect accepted those recommendations in the 1980 revision, although litigation was required to confirm definitively that computer software was protected by copyright. See Arthur R. Miller, Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since contu, 106 Harv. L. Rev. 977, 978-980 (1993); Paul Goldstein, Copyright § 2.15.2n. 46 (1989). See Williams Electronics, Inc. v. Artic Int’l, Inc., 685 F.2d 870, 875 (3d Cir. 1982); Apple Computer, Inc. v. Franklin Computer, Inc., 714 F.2d 1240, 1248-1249 (3d Cir. 1984).

[6] See discussion infra notes 156-174 and accompanying text.

[7] These issues will be discussed at length below.

for software and that if the Congressional mandate to protect software through copyright cannot be reversed, then at least that protection should be as sharply limited. Attacks on copyright protection sometimes emphasize its useful and indeed functional character.[8] In doing so, critics sometimes overlook the historical reality that copyright from the beginning has applied to books even though they may be highly functional—dictionaries, for example—and to other highly useful writings such as maps and charts.[9] Copyright protection has been accorded to the output of several new, functionally useful technologies (notably photographs in 1865).[10] In contrast, artistic works were not protected at all in the beginning and were first protected by a 1802 amendment, but only to the extent of designs, prints, etchings and engravings, categories of artistic works with pronounced technological aspects.[11] Indeed, it was only in 1870 that paintings, sculpture and similar more purely artistic works were accorded protection.[12] Thus, it can be said that the utility and functionality of works has been at least as much an incentive as a hindrance to Congressional inclusion of a category of works within the sphere of copyright protection.

It is well established that the fact that a writing is useful and functional does not preclude its protectability as a literary work and at most limits

[8] See Pamela Samuelson, Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs, 70 Minn. L. Rev. 471, 475-476, 507-511 (1985) and contu Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form, 1984 Duke L.J. 663, 741-753; Gary R. Ignatin, Let the Hackers Hack: Allowing the Reverse Engineering of Copyrighted Programs to Achieve Compatibility, 140 U. of Pa. L. Rev. 1999-2000 n. 17, 2021 (1987). See also Statement of Issues Presented to Conferees at the LaST Frontier Conference on Copyright Protection of Computer Software, 30 Jurimetrics 11, 18-19 (1989).

[9] The first copyright statute granted protection to “any map, chart, book or books.” 1 Stat. 124 (1790). See Blunt v. Patten, 3 Fed. Cases 763 (S.D.N.Y. 1828) (copyright upheld for navigational charts); and David P. Currie, The Constitution in Congress: Substantive Issues in the First Congress, 1989-1791, 61 U. of Chi. L. Rev. 775, 827 (1994) (“a rather generous if appropriate interpretation of the constitutional term ‘writings’....[T]here was no *requirement* in the copyright law that the author’s work be useful....” See also Jane C. Ginsburg, Creation and Commercial Value: Copyright Protection of Works of Information, 90 Col. L. Rev. 1865, 1873-1881 (1990), concerning the emphasis on informational works in the first century of U.S. copyright.

[10] 13 Stat. 540. See summary of various copyright amendments in Robert A. Gorman and Jane C. Ginsburg, Copyright for the Nineties 7-13 (4th ed. 1993), and see Ginsburg, supra note 9 at 1917.

[11] This 1802 statute gave copyright protection to whomever “shall invent and design, engrave, etch, or work...any historical print or other print or prints.” 2 Stat. 171.

[12] 16 Stat. 198, 212. Musical compositions were added in 1831. 4 Stat. 436.

the breadth of protection.[13] Nonetheless, some critics seem to hold the implicit view that copyright should be reserved for the finer things in life, such as poetry and creative prose—as well, of course, as academic writings. Protecting software as a literary work seemed particularly galling to some critics since in some forms software cannot be read by humans, since it is intended for "reading" by machines, and since it may someday actually be written by machines.[14]

The argument is occasionally heard that the usefulness of software makes it Constitutionally inappropriate for software protection because the Constitutional text indicates that "useful Arts" are to be protected by patent only. This position is based on the syntactical argument that the Constitutional text parses better when interpreted that way. The clause in question reads: "[t]o Promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive right to their respective Writings and Discoveries." On this argument Science-Authors-Writings are in one bucket and useful Arts-Inventors-Discoveries

---

[13] The Copyright Act of 1976 defines "useful article" but the function of the term is to limit one of seven categories of "works of authorship," namely, "pictorial, graphic and sculptural works," 17 U.S.C. § 102, whose definition provides that a useful article can be such a work "only if, and only to the extent that such design incorporates pictorial, graphic or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article." 17 U.S.C. § 101. There is no similar limitation for "literary works." Cf. Harper House, Inc. v. Thomas Nelson, 889 F.2d 197, 203 (9th Cir. 1989). And see Lotus Development Corp. v. Paperback Software Int'l, 740 F.Supp. 37, 55-58, 71-72 (D. Mass. 1990). The fact that software may be functional is only relevant, as Professor Goldstein points out, to the extent that its functionality imposes limitations on what is included within the scope of the copyright; to achieve functionality some parts will inevitably constitute ideas, procedures and other unprotectable elements under § 102(b), see Paul Goldstein, Copyright § 2.15 (1989), although the expression of those elements may be protectable.

[14] Protection as a literary work does not imply that software is literature. The House Report on the Copyright Act of 1976 made clear that the term "literary works" did not "connote any criterion of literary merit or qualitative value; it includes catalogs, dictionaries, and similar factual, reference or instructional works and compilations of data." H.R. Rep. No. 1476, 94th Cong. 2d Sess. 54. It went on to say that literary works included computer software only to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves. Ibid. Many forms of functional writings have long been protected; for example, in Reiss v. Nat'l Quotation Bureau Inc., 276 Fed. 717 (S.D.N.Y. 1921), Judge Learned Hand held that a code book containing 6,325 coined, meaningless words was entitled to copyright protection.

are in an entirely different bucket.[15] Uncritical acceptance of this syntactical argument could lead to the conclusion that the fact that something is a "useful Art" means that it may only be patented and not copyrighted.[16]

This syntactical argument does not, however, appear to comport with the usage of the Founders. For example, in 1783 the Continental Congress appointed a committee, including James Madison, to determine "the most proper means of cherishing genius and *useful arts*...by securing to authors or publishers of new books" a property right.[17] Not surprisingly, it was clear to Justice Holmes, speaking for the Supreme Court in *Bleistein v. Donaldson Lithographing Co.*, that the term "useful Arts" refers, among other things, to copyright and hence circus posters could be copyrighted.[18]

The syntactical argument reserving "useful Arts" to patents alone is further weakened by the inconvenient fact that "invention" was early used to refer not just to what was patentable but also what was copyrightable. The 1802 Act extending copyright protection to the output of a technology—there prints—provided that "he who shall invent...any historical or other print" shall be entitled to a copyright.[19] After copyright protection was extended in 1865 to another technology—photography—it was natural for the Supreme Court to refer to a photograph of Oscar Wilde as the "product of...intellectual invention" and therefore copyrightable.[20] Consequently, the Supreme Court today has no difficulty

[15] See Paul Goldstein, Copyright §1.01note 1 (1989) and Infodeck, Inc. v. Meredith-Webb Printing Co., 830 F.Supp. 614 (N.D. Ga. 1993). Contra: Lyman Ray Patterson, Copyright in Historical Perspective 193 (1968) and three recent Supreme Court cases cited infra note 21. See also William F. Patry, Latman's The Copyright Law 17 n. 4 (1986).

[16] Even if one were to accept this syntactical argument, software would of course still be protected under the "Science-Authors-Writings" alternative. The point of the analysis in the text is simply to show that an attack on copyright protection for software based on its utility misconstrues the Constitution.

[17] William Winslow Crosskey, Politics and the Constitution, Vol. I, p. 484 (1953). (Emphasis added). A 1786 North Carolina copyright statute made clear that among the purposes of copyright was "to promote useful discoveries." Library of Congress, Copyright Enactments 15 (1963). Patterson, supra note 15 at 187.

[18] 188 U.S. 239, 249 (1903).

[19] 2 Stat. 171 (emphasis supplied). See discussion of the 1802 Act supra note 11and accompanying text. See also the use in 1782 of the word "invention" in connection with copyright protection for books intended for primary school education in Jane C. Ginsburg, A Tale of Two Copyrights: Literary Property in Revolutionary France and America, 64 Tulane L. Rev. 991, 1000 (1990).

[20] Burrow-Giles Lithographic Co. v. Sarony, 111 U.S. 53, 60 (1884) Accord: Yuengling v. Schile, 12 Fed. 97, 100 (1882) (only "authors and inventors" entitled to copyright). In

in repeatedly stating that the purpose of copyright law includes the promotion of both “Science” and “useful Arts.”[21]

The conclusion that the usefulness of software is an argument for, rather than against, copyright protection of software is thus fully justified not just by the history of the copyright laws but also by the text of the Constitution when read in the light of the usage of the Constitutional terms by the Founders at the time of the drafting and by both the Congress and the Supreme Court in the century that followed.[22]

II. A Brief History of Computer Practices and Terminology

A third, and reinforcing, level of controversy has to do with the software industry itself. This critical perspective, which emphasizes the special nature of that industry and the sociology of software writers, has burst into prominence with the rapidly growing popularity of personal computers. The idea has spread that consumers and the software industry itself would benefit if popular software products could be readily replicated by competitors. Under this view broad copyright protection is bad policy to the extent that it reduces the ability to replicate such products. As we shall see, this idea is often discussed under the rubric of compatibility and de facto standards.

Before proceeding to the issues, it is worth noting that this particular mindset with respect to software had its historical roots in the computer hardware industry. Indeed, it is remarkable how the ideas advanced for narrow software protection parallel longstanding popular ideas about competition in hardware.

The origins of these notions can be traced to the commanding position that ibm acquired, after an initially slow start, in the computer industry of the late 1960s and the 1970s.[23] At first ibm and other computer manufacturers did not sell software; the buyer bought a mainframe and

Burrow-Giles, Justice Miller noted that the 1790 and 1802 statutes represented, in effect, an interpretation of the Constitution by “members of the convention which framed it” and speculated that photographs would have been included if only photography had been discovered at that early date. 110 U.S. at 57-58.

[21] Campbell v. Acuff-Rose Music, Inc., 114 S. Ct. 1164, 1169 (1994) (“copyright’s very purpose, ‘[t]o promote the Progress of Science and useful Arts’”); Feist Publications v. Rural Telephone Service Co. 499 U.S. 340, 349 (1991). (“The primary objective of copyright is...[t]o promote the Progress of Science and useful Arts.”); and see Fogerty v. Fantasy, Inc., 114 S. Ct. 1023, 1029 (1994).

[22] See also Ginsburg, supra note 9 at 993-994.

[23] On ibm’s slow start, see Thomas J. Watson, Jr., 227-230 (1990).

the software was “bundled” with it.[24] But even after “unbundling” at the end of the 1960s,[25] a commercial fact of life was that sellers of hardware peripherals, such as printers and storage devices, had to align themselves on ibm’s own proprietary standards to be successful. For a variety of reasons, it became conventional for peripherals to become ibm “plug compatible”[26] and for new application software to be written to run on ibm machines and operating systems.[27] Along with those developments came the phenomenon of the “compatible” processor, which rather than being intended to work with an ibm mainframe in the manner of a peripheral device, was intended to substitute for an ibm mainframe.[28] With the advent of the ibm personal computer, compatible computers became known as clones, no doubt to emphasize their essential identity with ibm machines.[29]

---

[24] Franklin M. Fisher, James W. McKie and Richard B. Mancke, ibm and the U.S. Data Processing Industry: An Economic History 23-25 (1983).

[25] Id. at 176-179.

[26] Derek F. Abell, Defining the Business 29-57 (1980). See California Computer Products, Inc. v. Int’l Business Machines Corp., 613 F.2d 727, 731 (9th Cir. 1979), where the court pointed out that the plaintiff peripheral manufacturers’ policy was to “copy and, where possible, improve upon an ibm design, and undersell ibm to its own customers.” The plaintiff “was able to avoid ibm’s expenditures for research and development and pass the savings on through lower prices” by “simply buying a device from ibm, taking it apart, and building a similar one.” Ibid. But ibm asserted no intellectual property rights in the products, and so the issue was whether ibm’s competitive responses violated the antitrust laws. The California Computer and other courts held for defendant ibm on the antitrust issues. See, e.g., Telex Corp. v. Int’l Business Machines Corp., 510 F.2d 894 (10th Cir. 1975).

[27] Katherine Davis Fishman, The Computer Establishment 232-234 (1981).

[28] Id. at 243-244. And see Lawrence A. Sullivan, Monopolization: Corporate Strategy, the ibm Cases, and the Transformation of the Law, 60 Texas L. Rev. 587, 599-601 (1982).

[29] Charles Ferguson and Charles R. Morris, Computer Wars 51-53. As in the case with mainframes, ibm was late to the personal computer market. Apple Computer, one of the first, was able however to fend off clones. See G. Christian Hill, Apple Refocuses Push to License Its Mac Program, Wall Street Journal B6 (July 18, 1994, characterizing “Apple’s failure in the late 1980’s to license its Mac operating system [as] one of the great strategic errors of modern business history.” For a variety of antitrust and business reasons, ibm adopted what turned out to be an “open systems” approach to microcomputers: “The specifications of ibm’s pc were easily obtainable, allowing independent hardware companies to make compatible machines and independent software vendors...to write applications that would run on different brands.” Harvard Business School, Apple Computer 1992, p. 3 (Case 9-792-081) (1993). After a time Apple became the principal personal computer manufacturer to pursue its own proprietary standards, and most of the rest of the industry began to produce ibm clones. Ibid; Ferguson and Morris, supra at 136-137. The word clone was originally used to refer to the

The concept of the clone became so widely accepted that it seemed to be an intrinsically inevitable aspect of the computer industry. And if clone machines were in the nature of the industry, then why not clones of operating systems? More generally, why not replicate application programs, whether they be games,[30] publishing programs,[31] or spreadsheets.[32]

The leap from replication of processors and operating system software to replication of application software products of smaller computer software firms can be seen in retrospect as a major transition for the software industry. When ibm was the target, defenders of the practice could refer to ibm's strong market position, and especially its large installed base, which made newcomers feel disadvantaged. The focus from the beginning with application programs, however, was replication, with comparatively little interest in attaching one application program to another. The reason was that, in general, little money was to be made attaching one application program to another. Rather, the financial payoff came with replicating a product that had already achieved mass popularity in the marketplace.

Before turning to the idea of replication of application programs, it is important to examine a related phenomenon, which is the use in one program of a particular convention that has proved useful in another program for invoking a specific computer step. The importance of such industry conventions in improving the ease of using a computer is most easily seen in a set of developments originating with the Apple Macintosh computer. The well-deserved reputation of Apple for ease of use stemmed in large part from the early insight of Apple hardware and software designers that users should be able to take the same action (pointing with a mouse to the same icon or using the same keystroke) to engender the same computer response no matter what software application they were using.[33] The widespread, even universal use of these conventions for

most inexpensive machines. Increasingly, however, manufacturers of higher end personal computers also aligned themselves on ibm proprietary standards so that their machines could use software written to those standards.

[30] See, e.g., Williams Electronics, Inc. v. Artic International, Inc., 685 F.2d 870 (3d Cir. 1982).

[31] See, e.g., Broderbund Software, Inc., v. Unison World, Inc., 648 F.Supp. 1127 (N.D. Cal. 1986).

[32] See, e.g., Lotus Development Corp. v. Paperback Software Int'l, 740 F.Supp. 37 (D. Mass. 1990).

[33] For example, the same key or combination of keys is, in principle, used to perform the same operation in every Macintosh-based application. This result is facilitated in part

application programs for Apple computers contrasted with the experience of many ibm and ibm clone pc users who found that the keystrokes for each pc application program had to be learned one program at a time. The result was that programmers began to use these Apple conventions even for pc-based application programs.[34]

Adopting a particular convention originated by another for invoking a particular response from a computer is quite a different thing from the replication of an entire computer program. Such replication is a difficult task. Difficult, that is, unless one simply copies a successful application program. But outright copying was prohibited by the 1976 Act and the 1980 amendment affirming copyright protection for computer software.[35] As a result follow-on competitors had two alternatives. One was to start by copying the software of the innovator but then manipulating the programming in such a way as to make detection of copying more difficult.[36]

The alternative was to write a program independently but to try to come up with a product that looked and felt to the user essentially the same as the target program. The most important thing for follow-on firms to copy if they were to take a ride on the popularity of a well-known application program was the way the program looked to the human user and the way the user interacted with the program. Since the typical user cared little about the underlying programming (which was invisible to him), it became popular to refer to this external aspect of an application program as its *user interface*.[37]

---

by that portion of the Macintosh operating system known as the Macintosh Toolbox, with which all program written for the Macintosh are forced to interact. "It's said that once you learn one Mac program, you know how to run any Mac program. This is no coincidence: It's built into the system." John Rizzo and K. Daniel Clark, How Macs Work 25 (1993). See Apple Computer, Inc. v. Microsoft Corp., 799 F.Supp. 1006, 1019 (N.D. Cal. 1992).

[34] One aspect of this cross-fertilization can be seen in the Apple-Microsoft litigation. See Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1017-1022 (N.D. Cal. 1992).

[35] See supra note 5.

[36] See Miller, supra note 5 at 1027, and sources cited therein, with regard to disguising traces of copying.

[37] The concept of the user interface is not well-defined. However, a joint pto/Copyright Office document states that the term is commonly limited to six elements: commands, menus, questions and answers, form filling, icons, and function keys. U.S. Patent and Trademark Office and U.S. Copyright Office, Patent-Copyright Laws Overlap Study 44-45 (May 1991) ("Joint Overlap Study").

The copying of the user interface became the center of the controversy. While most observers were prepared to agree that outright copying of the underlying programming was infringement, some nevertheless defend the copying of "interfaces," meaning thereby the user interface. This line of argument rests on a facile and misleading use of the term "interface" to refer to two fundamentally different things. In normal circumstances involving software copyright the word "interface" is used, for example, to justify attachment of one software product to another software product. But when speaking of the "*user* interface" the term is utilized to justify the replacement of the latter. More specifically, the notion that copying hardware and software interfaces for the purpose of promoting attachment was not piracy but rather was efficiency-promoting through facilitating interconnection thereby provided a verbal formula for justifying outright copying of application software. If one could argue that copying interfaces was defensible, why not the user interface?

Comparing the user interface to a hardware/hardware or a software/hardware or a software/software interface was thus a tendentious apples and oranges comparison. The user is not "attached" to the application program in the manner that peripherals are attached to hardware, or application programs to operating system programs, or even one application program to another. Still, the term "user interface" caught on because the user did interact with a program; the idea of a user interface consequently became accepted. In the jargon-filled world of computer-speak, software firms began to advertise their user interface. When icons became popular, for example, firms vaunted their graphical user interface ("gui").

Powerful commercial motives for the follow-on companies to make the user interfaces as close as possible to the innovator's user interface became conjoined with two phenomena that together generated sharply differing views between the commercially successful first-generation application software companies and their follow-on competitors. The first was that independent replication, even of user interfaces, was difficult, costly, and often not successful—at least without access to the target program code. And the second was that the culture of the software community tended to view existing software products much like academics view library books—something to be acquired, studied in detail and then borrowed from.[38] The result of carrying this attitude too far was sometimes the

[38] Cf. Pamela Samuelson and Robert J. Glushko, Comparing the Views of Lawyers and User Interface Designers on the Software Copyright "Look and Feel" Lawsuits, 30

software equivalent of academic plagiarism—that is, the copying of the programming itself.

Against this background of commercial opportunity, difficulty of replication, and a culture that built on and freely borrowed from the work of predecessors, three separate notions developed that supported the inclinations of the follow-on software firms: that software is not sufficiently creative to deserve copyright protection, that the public would be better off with greater choice among application software programs, even if the programs were substantially identical, and that software is simply so "different" that it requires its own form of protection. These three notions crystallized in a few quite specific criticisms of the evolving law insofar as it served to protect the first-generation innovating firms.

The first criticism is that since copyright protects only expression and not ideas, the courts should adopt an approach that gives an expansive interpretation to the concept of ideas and a narrow interpretation of expression.[39] The second criticism is that where software, whether operating system or application, becomes so highly successful that it becomes a "de facto standard" to which newer firms must adjust in order to be commercially successful, then the constraints on actual copying should be relaxed. The third criticism is that neither copyright nor patent law is the appropriate basis for protection and that a *sui generis* form of protection should be substituted.

With regard to the first criticism, the idea/expression dichotomy raises few economic issues unless the courts shrink the area of protected expression too far to permit the copyright law to deal adequately with the fundamental problem that intellectual property is designed to solve—the appropriability problem.[40] Indeed, that is arguably what happened in at

Jurimetrics 121, 136-137 (1989): "When [software engineers] see good ideas and the research that lies behind them, they feel they can incorporate these designs into new products of their own. They do not consider themselves thieves, plagiarists, or copyright infringers when they do so, but rather they consider themselves scientists and engineers who are innovating on top of others' ideas in the kind of evolutionary fashion which has exemplified development in this fields." For an illustration of this software industry attitude, see Michael J. Miller, Software Patents Must Go, pc Magazine 79 (March 15, 1994).

[39] A variant of this criticism emphasizes not so much the unprotectability of ideas but the unprotectability under 17 U.S.C. § 102(b) of "any...procedure, process, method of operation, concept, principle or discovery."

[40] For a discussion of the appropriability problem, see infra notes 45-51 and accompanying text. Economic problems may equally be raised if copyright is extended so far that future innovation is stymied. See notes 62-65 and accompanying text.

least one recent case, in the service of dubious economic arguments.[41] Alternatively, the idea/expression dichotomy could raise economic issues if the area of protected expression were expanded too far into the area of ideas, since the effect would be to discourage unduly future innovation. The second and third criticisms, however, do raise economic issues. In fact, the second criticism (which is variously referred to under the headings of compatibility, de facto standards and network externalities) explicitly relies on economic ideas. The third criticism, seeking to justify a sui generis approach, is not primarily about economics but does raise one significant economic issue.

These three central lines of copyright argument have been advanced with great vigor, learning, and resourcefulness. But the issue, especially with respect to the last two criticisms, is whether they represent sound policy. That question can be answered at a number of levels. Here I do not propose to spend much time on the legal conceptual level, which has been analyzed at great length, usually from a traditional copyright point of view. Although some brief reference to the doctrinal issues is necessary, my focus will be on whether economics has anything useful to contribute to the debate.

After dealing with the copyright issues, I shall examine the software patent issues from an economic perspective. As we shall see, the economic analysis of issues involving software-related patents is not a fundamentally different inquiry from that involving copyright. Nevertheless, the patent issues, coming from a different intellectual property tradition, are necessarily different at the doctrinal level, as I shall explain. Finally, I shall deal briefly with the relative merits of a sui generis approach, again largely from an economic point of view.

III. A Survey of the Economics of Intellectual Property

As already emphasized, software is, among other things, a “useful Art” with profound commercial and economic significance. Innovation is, of course, the heart and soul of patent protection, but some people apparently find it rather strange to think of copyright as protecting innovation. Yet not only contu but the Congress as well determined that copyright should be utilized to protect software innovations.[42] It is therefore not only warranted but, if one takes the legislation seriously,

[41] See discussion of the Computer Associates case, infra at notes 74-85 and accompanying text.

[42] See discussion of new technology in contu 26.

necessary to look to the economic aspects of intellectual property law generally in determining how copyright principles should be applied in carrying out that Congressional mandate.

Copyrights, like patents, are best thought of as property rights. The property character of patents is explicit in patent law[43] and only slightly less so in copyright law.[44]

The fundamental justification for creating property rights in the results of innovation is to deal with the appropriability problem.[45] This problem arises from the simple circumstance that in the absence of an intellectual property right, the innovator would not be able to recoup his research and developments costs in competition with those who, thanks to copying, were able to sell the product without incurring those costs; therefore, the rate of r&d would be less than it would be with an intellectual property right and specifically, from an economic point view, that rate would be less than optimal. This appropriability point, which refers to the inability of the innovator to appropriate the benefits of his own r&d investments, is obvious in the case of patents,[46] but it is equally true in the case of copyrights, even where the copyright covers works of such a purely literary or artistic nature that the concept of r&d costs may seem far-fetched. Using books as an example, Landes and Posner speak of the expected "cost of expression," which must be recoverable if new works are to be created. In the absence of a property right, they point out, "The market price of the book will eventually be bid down to the marginal cost of copying, with the unfortunate result that the book probably will not be

[43] 35 U.S.C. § 261. See Kenneth W. Dam, The Economic Underpinnings of Patent Law, 23 J. Legal Stud. 247, 253 (1994) (hereafter Dam, Economic Underpinnings).

[44] Copyrights are, as common legal parlance indicates, intellectual property. The Copyright Act defines "ownership," 17 U.S.C. § 201, and the fact that the ownership of a copyright is separate from ownership of any particular copy underscores the intangible property character of a copyright. Donald S. Chisum and Michael A. Jacobs, Understanding Intellectual Property Law 4-184 n. 98 and 4-266. For a detailed analysis of the property character of copyright, see Wendy J. Gordon, An Inquiry into the Merits of Copyright: The Challenge of Consistency, Consent, and Encouragement Theory, 41 Stan. L. Rev. 1343, 1354-1394 (1989).

[45] Dam, Economic Underpinnings 247.

[46] See the explanation of the appropriability problem in the patent context, ibid. The term "appropriability" can also be used to point out the ability, in the absence of an intellectual property right to "appropriate" the benefits of the investment in innovation of others—to reap where one has not sown.

produced in the first place, because the author and publisher will not be able to recover their costs of creating the work."[47]

Although the notion of creating a property right to deal with the problem of appropriability is, with limited exceptions,[48] uncontroversial with respect to such core subject matter as copyright in books, it is sometimes argued that copyright protection is not necessary for software. The fact is, of course, that Congress chose to cover software just as it did the output of other technologies in the past such as engraving and photography.[49] In the face of that clear Congressional choice, those who believe that protection is unnecessary to deal with the appropriability problem with respect to computer software are reduced to essentially two conclusions: that Congress was wrong and therefore the legislative choice should be reversed or, alternatively, that the courts should construe the legislative choice as narrowly as possible to eliminate copyright for software in particular instances, say with respect to user interfaces. Before returning to the user interface problem, it is useful to consider the first of the foregoing two arguments, namely, the preliminary question whether copyright is necessary at all for software.

If copyright is conceded to be necessary for the other subjects of copyright, including those with a technological basis, then surely the burden shifts to explain why there is no appropriability problem with respect to software. One possibility lies in the frequently heard assertion that being first to the market is a sufficient incentive for the writing of software. Indeed, it is sometimes said that the first to market obtains such an advantage that subsequent superior products have a special disadvantage due to what are sometimes called "switching costs" (or the equivalent, "lock-in costs").[50] Despite the undeniable existence of some switching costs where a particular program becomes well-established in the marketplace, one must remember that with modern technology the cost of copying software is trivial—essentially the cost of the media—and

[47] William M. Landes and Richard A. Posner, An Economic Analysis of Copyright Law, 18 J. Legal Stud. 325, 328 (1989). See also William M. Landes, Copyright Protection of Letters, Diaries, and Other Unpublished Works: An Economic Approach, 21 J. of Legal Stud. 79, 82-83 (1992).

[48] See reservations in Stephen Breyer, The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs, 84 Harv. L. Rev. 281, 291-308 (1970). But see Barry W. Tyerman, The Economic Rationale for Copyright Protection for Published Books: A Reply to Professor Breyer, 18 U.C.L.A. L. Rev. 1100 (1971).

[49] See discussion supra at notes 4-92 and accompanying text.

[50] See the discussion of switching costs infra at notes 95-100 and accompanying text.

that the time it takes to copy is far less than in the case of most other subjects of copyright. In short, the advantages of being first to market, far from being a substitute for copyright, are themselves a function of copyright, because follow-on firms must themselves innovate under a copyright regime, and first-to-market advantages would largely disappear if copyright were eliminated for software. The first-to-market argument against software copyright thus proves to be a chimera when one approaches it more closely.[51]

Another line of argument against copyright is that many users will choose not to copy but rather will acquire software in a legitimate market transaction in order to have ready access to upgrades, manuals, help lines and similar features of modern software marketing. This argument is directed, however, to only one of the three common types of copying, namely, the making by users of extra copies beyond those permitted by contract or “shrink-wrap” license. Even assuming the merits of the argument in that limited sphere, it has no merit in the context of the two other types of copying that currently plague software innovators: *piracy*, in which the copier sells a copy as an original and *me-too copying*, in which copying is used to make me-too, follow-on products for sale under the copier’s own trade name. Piracy is almost universally thought to be worthy of legal condemnation, except perhaps in some developing nations where some seek to justify it on poverty or development grounds and where powerful interest groups have grown up to protect piracy of software, tapes, records and movies.[52]

It is the third form of copying—in order to sell me-too products in competition with a popular innovation—that has in fact given rise to the most litigation and where, as we shall see, rather sophisticated arguments have been advanced to justify the practice. These arguments, based on compatibility, de facto standards and network externalities, will be considered at length below. But even if one were to accept these specialized arguments, they at best would lead to making exceptions to copyright protection in quite limited situations, not to abolishing copyright protection for software.

In considering the preliminary question whether copyright protection should be available at all for software, it is worth bearing in mind that

[51] A further and related question involves whether protection is needed for first-to-market software programs that become industry standards. See discussion infra notes 125-155 and accompanying text.

[52] Piracy normally involves passing off and therefore violates trademark law as well as copyright.

some arguments that have been advanced against the entire institution of copyright actually have less force for software than, say, books.[53] For example, it is sometimes said that books would not disappear with the abolition of copyright because authors have many motives ranging from egotism to self-promotion to a quest for immortality and hence the need for economic incentives to deal with the appropriability problem is not such an important factor as one might suppose. Whatever the merits of this argument for books, it has less merit in the modern software market where much of the writing of software is a faceless, team effort without such non-monetary incentives and, in any event, many modern software products are so complex that they will only be undertaken by commercial firms operating with commercial incentives.[54]

To say that a property right in software is necessary to deal with the appropriability problem is not to say that no economic problems are created by the use of copyright. But these economic concerns can be better assessed once we accept that the appropriability problem must be solved by some form of property rights if the necessary economic incentives are to be created for software innovation. The three principal economic concerns—monopoly, rent seeking and the possibility that innovation today may be unduly favored over innovation tomorrow—warrant brief treatment.[55]

The monopoly concern is overstated for two reasons. First, simple observation tells us that even patents (which create a legal right to exclude even in the case of independent creation) generate few monopolies in the market sense; large r&d-oriented firms generate hundreds of patents a year and yet few such firms have monopolies in any economic market.[56] If this is true of patents, it seems even clearer in the case of copyrights where no power to exclude is granted, where only the power to preclude copying is granted, and where independent creation by competitors is a

[53] Landes and Posner, supra note 47 at 331.

[54] Even authors of “shareware,” which is distributed free or for a minor administrative charge, normally rely on copyright. See Anthony Lawrence Clapes, Softwars 142 (1993). Nonetheless, shareware does exist and may be quite sophisticated, and the writers of shareware do not normally try to enforce their rights.

[55] The three concerns are analyzed at length in the patent context in Dam, Economic Underpinnings. In the copyright context the terms used to refer to the need to avoid unduly discouraging future innovation vary widely by author and subject, but include “access,” “distribution,” “exploitation,” and even “competition.”

[56] Id. at 249-250.

complete defense.[57] Second, according copyright protection does not permit the innovator to restrict production, the hallmark of monopoly. At most, the innovator will capture economic rent at the same level of output as existed in the market before the innovation and, in the case of major innovations leading to sharply reduced costs, output may actually expand.[58] In short, output will be the same or higher with the copyrighted innovation than without the innovation. To be sure, if we assume, as does most of the literature assumes at least implicitly, that the innovation would have been made without the intellectual property right (or would have been made by someone else very soon), then it may make more sense to talk about monopoly and restriction of production. The question is which perspective provides the most insight into the role of intellectual property rights. As I have argued elsewhere, it makes more sense, and better illuminates intellectual property law, to emphasize the rent seeking rather than the monopoly problem.[59]

The fact that the grant of an intellectual property right may permit the innovator to enjoy economic rent suggests that rent seeking may be a problem. Economic rent is in one sense just a name that we give to the incentive accorded the innovator by intellectual property protection, but still public policy should attempt to minimize unnecessary rent seeking. How important the rent seeking problem may be in the case of patents is

[57] Both the Federal Trade Commission and the Antitrust Division investigated a variety of public charges of monopoly and restraint of trade that had been made by private parties against Microsoft. In the end Microsoft entered into a consent decree with the Department of Justice. The consent decree was limited to Microsoft’s licensing practices. In the most important provision Microsoft agreed not to charge royalties to pc sellers on a per processor basis (which involved royalties even on machines loaded with a Microsoft competitor’s operating system) but rather to charge only for each Microsoft-installed operating system. See Microsoft Settles Accusations of Monopolistic Selling Practices, 67 BNA Antitrust and Trade Reg. Rep. 106 (July 21, 1994).

[58] See the explanation, using the example of patents, in Dam, Economic Underpinnings 250-251, especially note 9. As pointed out there, the fact that output may be lower than it would be if a patent were not granted and competitors could freely use the innovation should not obscure the point that a patent does not lead to a restriction of output below what it would have been in the absence of the innovation. Indeed, output will expand if the innovation is a major one. See Dennis W. Carlton and Jeffrey M. Parlor, Modern Industrial Organization 666-679 (1989), including Figure 20.3(b) and accompanying discussion.

[59] See Dam, Economic Underpinnings. For example, patent law deals effectively with the situation where the invention would have been made by someone else very soon anyway. It does so through the nonobviousness requirement, which renders patent protection unavailable for an invention that would have been obvious to someone skilled in the art. Id. at 259 n. 53.

an open question, though various patent doctrines reduce the extent of any economic rent.[60] In the case of copyrights, the fact that there is no power to exclude independently created works suggests that rent seeking is less of a problem than in the case of patents.[61] Moreover, the copyright doctrine of merger of expression and idea assures that there is no copyright protection where there are no (or very few) alternative means of expressing an idea and therefore underlying ideas and not just particular software expression would be engrossed. This merger doctrine not only tends to sharply reduce economic rent, but beyond that it also therefore tends to reduce rent seeking. In the absence of the merger doctrine, firms would especially seek to be the first to find the sole way of expressing a software idea since they could thereby exclude follow-on firms from using that idea, thereby carving out a zone of protection analogous to that accorded by a patent to an original technological concept.

The third, and most important problem, concerning copyright protection for software is the possibility that such protection will unduly favor innovation today at the expense of innovation tomorrow. An important function of an intellectual property regime should be to achieve an appropriate stream of innovation over time.[62] This is, of course, a problem for copyright generally, not just for software copyrights. Landes and Posner, proposing a concept of the "cost of expression," show that "beyond some level copyright protection may actually be counterproductive by raising the cost of expression" since the "less extensive copyright protection is, the more an author, composer, or other creator can borrow from previous works without infringing copyright and the lower, therefore, the costs of creating a new work."[63] As they point out, "[F]rom an ex ante viewpoint, every author is both an earlier author

[60] See generally Dam, Economic Underpinnings, 261-266.

[61] Independent creation is an important exception for literal code but may be less available where user interfaces are in question. In the case of popular mass-market programs, for example, it is unlikely that any software engineer capable of writing such a program would not previously have seen the screen displays or would be unfamiliar with the other user interface elements. From a legal point of view, it is plausible to assume access and therefore, if the original and competing user interfaces are substantially similar, to infer copying. Moreover, the clean room concept for creating competing software products (under which software engineers for follow-on firms work in the isolation of a "clean room" with only general specifications from outside) may work where literal code is involved but is therefore less likely to shield the results from infringement liability in the case of non-literal elements such as user interfaces.

[62] See Dam, Economic Underpinnings 253, 266-267.

[63] Landes and Posner, supra note 47 at 332.

from whom a later author might want to borrow material and the later author himself."[64]

This third economic issue is the central economic problem in the software copyright field. Since it is an especially fast moving technological field and since this new field is still fertile with innovations, the possibility that the pace of progress could be slowed by copyright protection cannot be ignored. Yet a failure to accord any protection would certainly slow progress, and even more immediately, since it would discourage software r&d expenditures. The objective of copyright law policy should therefore be to achieve an appropriate balance of innovation over time.

IV. An Introduction to Software Copyright Doctrinal Issues

The prime copyright doctrine that achieves the balance between innovation today and innovation tomorrow is the idea/expression dichotomy. Ideas are freely appropriable and hence future innovators may freely rely on ideas in others' software to create new software.[65] Only expression is protected. But even expression is, as previously noted, freely appropriable under the doctrine of merger if the underlying idea can be expressed in only one or very few ways; in such a situation the expression is said to merge with the idea.

Because of the importance of the idea/expression doctrine to the flow of innovation in software, it should be no surprise that it has justifiably become the main battlefield between first generation innovators and follow-on firms in particular software fields. After briefly reviewing these cases, I shall then confront the question whether additional concepts are required, such as the concepts of compatibility, de facto standards and network externalities.

Software copyright idea/expression cases may conveniently, though somewhat artificially, be divided into three overlapping generations.[66] In the first generation—roughly the 1980s—the courts simply gave effect to the 1976 Act and the 1980 amendment: software is protected even if it is

[64] Id. at 333.

[65] 17 U.S.C. § 102(b) provides: "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." (Emphasis supplied) The term "idea" is not defined and thus can be said to be a metaphor for what courts choose not to protect because the costs of according protection, particularly the cost to future expression, are deemed not worth the benefits in terms of the added incentive to present expression.

[66] The classification of cases into three generations is one that seems most convenient to the author; there are other classifications.

utilitarian in character.[67] The form of the software—whether object code, source code or microcode—does not matter so long as it is "fixed in any tangible medium of expression"[68] and infringement can be established without direct proof of actual copying so long as the defendant had access to the copyrighted code and the plaintiff's and defendant's program are substantially similar.[69]

The second generation of idea/expression cases dealt with the question whether copyright protection extended beyond mere copying of the literal code to so-called non-literal aspects of a computer program. The leading case of *Whelan Associates* established the proposition that copyright protection extends beyond "programs' literal code to their structure, sequence and organization."[70] Cases of this second generation, which have sometimes mistakenly been referred to as "look and feel" cases,[71] have protected screen displays and other elements of the user interface without insisting on a showing that the underlying literal code was identical or even substantially similar so long as there are alternative forms of screen displays or other user interface elements to express the underlying idea.[72] This second generation has extended well into the 1990s. For example, in a series of decisions finding infringement of the Lotus 1-2-3 spreadsheet program, the screen display's "menu structure, taken as a whole—including the choice of command terms, the structure and order of those terms, their presentation on the screen, and the long prompts" were protected.[73]

The third generation began in the 1990s and overlaps in time the more recent cases of the second generation. In fact, the second and third generation precedents co-exist and are in different Federal judicial circuits; thus, the Supreme Court may have to determine which generation of cases

---

[67] See, e.g., Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983).

[68] 17 U.S.C. § 102(a); NEC Corp. v. Intel Corp., 10 U.S.P.Q. 2d 1177 (N.D. Cal. 1989).

[69] Atari, Inc. v. North American Philips Consumer Electronics Corp., 672 F.2d 607 (7th Cir. 1982); E.F. Johnson v . Uniden Corp. of America, 623 F. Supp. 1485 (D. Minn. 1985).

[70] Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (1986).

[71] The "look and feel" phrase is imprecise and therefore inappropriate. See the comments of Judge Keeton in Lotus Dev. Corp. v. Paperback Software Int'l, 740F. Supp. 37, 62-63 (D. Mass. 1990).

[72] See Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 809 (N.D. Cal. 1986).

[73] Lotus Development Corp. v. Paperback Software Int'l, 740F. Supp. 37, 68 (D. Mass. 1990).

survives. In these third generation cases, which also involve non-literal program components, some courts have become worried that protection was being too broadly accorded and have used a variety of subsidiary copyright doctrines to sharply reduce the realm of protectable expression and greatly expand the realm of unprotectable ideas in any program.

The leading case of this genre, the Second Circuit's *Computer Associates* decision, illustrates a methodology to accomplish this result.[74] First, it adopted a three-step procedure, involving "abstraction" and "filtration" of the copyrighted program and then, and only then, "comparison" of the two programs.[75] The abstraction step, taken from a famous Learned Hand dictum involving an alleged infringement of a play, "Abie's Irish Rose," by a motion picture, "The Cohens and the Kellys,"[76] emphasizes the need to seek out unprotectable elements at a number of levels of abstraction in the copyrighted program.

The filtration step involves the actual separation of unprotectable ideas from protectable expression at each of these levels through three successive techniques. The first applies the merger doctrine and broadens it to include a situation where the copyrighted program has chosen the most efficient manner of writing the code to accomplish a function; under this technique expression merges with the idea when the expression chosen is the most efficient, even though not the only way or perhaps not even one of a limited number of ways of expressing an idea.[77] The second technique, by analogy to the scenes a faire doctrine,[78] then excludes from

[74] Computer Associates Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992). Unlike most of the software copyright cases, the challenged program was an operating system program—a translator—rather than an application program. Moreover, it was designed to run on a mainframe, unlike most recent cases, which have involved software for personal computers.

[75] The Computer Associates three-step approach was partly adopted in Gates Rubber Co. v. Bando American Inc., 9 F.3d 983 (10th Cir. 1993), although the Gates court approved conducting the comparison step first, on the ground that copying of unprotectable elements is probative on the key factual issue of copying of protectable elements; only the latter, the court said, can give rise to liability. A much less elaborate filtration approach had been adopted before Computer Associates in Brown Bag Software v. Symantec Corp., 960 F.2d 1465 (9th Cir. 1992).

[76] See Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1931).

[77] 982 F.2d at 707.

[78] The scenes a faire doctrine, most easily understood in the fiction context, renders unprotectable stock "incidents, characters or settings which are as a practical matter indispensable" (such as, in fiction, a boy meets girl, boy loses girl plot) and thereby, in the software context, treats stock or conventional means of expression as ideas. See Atari v. North American Philips Consumer Elec. Corp., 672 F.2d 607, 616 (7th Cir. 1982).

infringement liability programmers' choices "circumscribed by extrinsic considerations."[79] The third technique is to exclude from the allegedly infringed program material taken from the public domain.

The final of the three steps, after abstraction and filtration, is comparison. The allegedly infringing program is compared with the now sharply truncated copyrighted program, from which have been "sifted out all elements...which are 'ideas' *or* are dictated by efficiency *or* external factors, *or* taken from the public domain."[80]

The *Computer Associates* case is particularly interesting for several reasons. From a legal point of view, it betrays a fundamental underlying disagreement with the Congressional decision mandate making copyright a means of protecting software. The opinion by Judge Walker explicitly states that the purpose of the three-step dissection is to assure that only a remaining "core" or "golden nugget" of expression is protected.[81] At the same time he overlooks the well-established proposition that a work composed entirely of unprotectable separate elements must nonetheless be protected if those elements are combined in a sufficiently original fashion.[82] Even more telling, the opinion forthrightly characterizes judicial copyright infringement analysis of computer programs as an "attempt to fit the proverbial square peg in a round hole" and calls for "further legislative investigation—perhaps a contu ii."[83]

More interesting for present purposes than the legal methodology by which *Computer Associates* sought to limit the availability of copyright protection are the assumptions underlying that methodology. While conceding that critics "have a point" in predicting that the methodology of

---

[79] 982 F.2d at 709. Among these extrinsic circumstances are "(1) the mechanical specifications of the computer of which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry." 982 F.2d at 709-710.

[80] 982 F.2d at 710 (emphasis supplied).

[81] 982 F.2d at 710.

[82] See discussion of this principle with illustrations, infra note 134. This principle is a specific application of the more general principle, articulated by Judge Learned Hand, that it is the sequence of unprotectable elements that constitutes protectable expression. Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936). Judge Walker defended his views against criticisms similar to those in the text, arguably modifying his position to come closer to the principle articulated in the text, in his comments at a symposium. Copyright Protection: Has Look & Feel Crashed?, 11 Cardozo Arts & Ent. L.J. 721, 726, 730 (1993).

[83] 982 F.2d at 712.

his decision "will be a disincentive for future computer program research and development," Judge Walker opines that the purpose of copyright law is not to confer a monopoly but rather to reward "artistic creativity."[84] Moreover, Judge Walker appears to misconceive the nature of innovation and the property rights justification for its protection by equating innovation with "hard work" and "significant labor and expense."[85]

V. Compatibility, De Facto Standards and Network Externalities

The three generations of software copyright cases have been analyzed without reference to a strong set of arguments that are to be found in the secondary literature. In addition to seeking to expand the realm of ideas and contract the realm of expression, academic critics have developed a line of thought that emphasizes the importance of computer software programs working together. The argument sails under a variety of flags. One variant stresses the importance of compatibility, another the values and dangers of standardization, a third, explicitly relying on the language of economics, stresses network externalities. These three lines of attack are at base one and the same argument.

1. Compatibility

In analyzing these three variants, a convenient point of departure is the notion of compatibility because it is the one that is most familiar in the computer industry. The idea of compatibility originally had more to do with hardware than software and dates back at least to the period when ibm emerged as the leading firm in the computer industry.[86] The notion of compatibility referred to a variety of specific industry issues. One had to

[84] 982 F.2d at 711. The virtually explicit assertion that copyright should only be available for "artistic" works must come as a surprise to holders of copyrights on non-fiction books, maps and the like. As the Supreme Court said as recently as 1991, "[I]t is beyond dispute that compilations of fact are within the subject-matter of copyright." Feist Publications, Inc. v. Rural Telephone Service Co., Inc., 499 U.S. 340, 345 (1991).

[85] Perhaps this passage in Judge Walker's opinion should not be taken too seriously since it is essentially a legal argument seeking precedential sustenance from the Supreme Court's decision the preceding year in Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340 (1991), rejecting the "sweat of the brow" doctrine in holding that purely factual compilations (there alphabetized telephone listings) were not copyrightable. Yet even Feist specifically held that the creativity required for copyright has nothing to do with the artistic creativity of which Judge Walker speaks and that even a "compilation of facts" is eligible for copyright if it features "an original selection or arrangement of facts." 499 U.S. at 350.

[86] The history of compatibility and allied notions in the computer industry is briefly summarized supra notes 23-32 and accompanying text.

do with whether ibm should be required to take steps to ensure that manufacturers of peripheral equipment would be able to attach their equipment; they could do so only if that equipment was “compatible” with an ibm mainframe. Indeed, the phrase to describe this relationship between ibm mainframes and non-ibm peripherals became “plug compatibility,” which meant essentially that a non-ibm peripheral could simply be plugged into an ibm mainframe. This sense of compatibility therefore stressed the ability to attach one machine to another where the machines did different things but by working together could achieve results not otherwise achievable. There was obviously a competitive element in the sense that with plug compatibility non-ibm peripheral manufacturers could compete with ibm peripheral products in sales to users who already owned one or more ibm mainframes.[87]

Compatibility soon took on, however, a second meaning. Competing mainframe manufacturers sought to make their mainframes compatible with those of ibm. The objective here was not to attach but rather to replace. In short, a compatible mainframe manufacturer could sell mainframes in competition with ibm to buyers who, because they had a large installed base of ibm mainframes, would not otherwise readily consider buying a non-ibm mainframe.

With the advent of microcomputers (pcs), a related concept grew up: the clone. The clone notion suggests an identity of product going beyond mere compatibility, but the idea of a clone was in essence the same as that of compatibility in the replacement sense. The clone could be sold to replace an ibm product (or indeed the product of another clone manufacturer).

We see then that from an early stage the concept of compatibility encompassed two rather different concepts: *(1) attachment*, in the sense that a compatible product of one kind—typically, a peripheral—could be sold to attach to a different kind of product—typically, an ibm mainframe; and *(2) replacement*, in the sense that a compatible product could be sold to replace a product of a similar kind. Of course, replacement as a concept need not mean literal replacement in the sense that the ibm mainframe would be taken out of service; rather it could mean simply that an ibm user could buy an additional mainframe from a second manufacturer without incurring additional costs (beyond the price of the additional mainframe) arising from the impact on the installed base of ibm

[87] For an important case from this era involving plug compatible competition, see Telex Corp. v. International Business Machines Corp., 510 F.2d 894 (1975).

mainframes. Indeed, the concepts of attachment and replacement converged linguistically as mainframes began to be networked in order to achieve greater computing power. Then the competing mainframe manufacturer could argue that his mainframe could "attach" to an ibm mainframe in the same way that two ibm mainframes could be attached to each other in a network.[88]

The distinction between attachment and replacement is essential to making sense out of the various arguments advanced in favor of compatibility in computer software. Some software attaches to other software. For example, application software attaches to operating system software. Similarly, some operating system programs attach to other operating system programs—for example, data base programs are sold by a number of firms in competition with one another and can be attached (if they are compatible) to, say, an ibm core operating system.[89] Again, one application program can attach to another application program if they are compatible.[90] And of course operating system programs, if they are written to be compatible with ibm mainframes, may be attachable not only to ibm mainframes but also to competing mainframes that are ibm-compatible in the replacement sense. For example, in a number of countries purchasers of mainframes from competing manufacturers nevertheless bought or licensed ibm operating systems to run on those "replacing" mainframes.

2. *De Facto Standards*

The term compatibility is closely related, in the arguments in favor of narrow protection for computer software, to the idea of standardization. Standardization in software initially emphasized the role of standards in affecting the ability of followers to attach to the software of those who came before. But as in the case of compatibility, it is essential for straight thinking to distinguish standardization to facilitate attachment from

[88] The phenomenon of the clone, whether mainframe or pc, also led to a convergence between attachment and replacement in particular cases. A clone processor could be viewed not only as a replacement but also as attaching to pre-existing peripherals. The point is that attachment is a two-way street. Processors attach to peripherals and vice-versa.

[89] See the description of an ibm mainframe operating system in Innovation Data Processing, Inc. v. International Business Machines Corp., 585 F.2d 1470 (D.N.J. 1984). Data base programs can be considered either operating system or application programs.

[90] See, for an illustration, Lewis Galoob Toys v. Nintendo of America, Inc., 780 F. Supp. 1283 (N.D. Cal. 1991), aff'd 964 F.2d 965 (9th Cir. 1992), involving entertainment software attaching to other entertainment software.

standardization to facilitate replacement. In addition, standardization is an overly blunt term that can refer to a variety of kinds of standards. One important, if ill-defined, distinction is that between de jure standards and de facto standards. The concept of de facto standards will warrant extensive analysis below because it has become the focus of an argument for limiting copyright protection, namely, that copying of a program should be permitted where it has become a de facto standard.

3. Network Externalities

Sometimes the ideas of compatibility and standardization are dressed up in the language of economics by referring to network externalities.[91] The notion is that where a network exists, social costs and benefits can rather easily and obviously differ from private costs and benefits. Computer software arguments thus draw sustenance from a well-known and accepted set of ideas on network industries such as railroads and telephones:

A node is a point in a network where two or more links intersect. A link is a path between nodes. Any action of a firm that affects one link in the network can affect the costs of all firms using links in the network. One firm’s actions can create an externality that will not be accurately reflected in a price system. That is, one firm’s actions can create costs that it does not bear, but that other firms do.[92]

One can easily see that computers are frequently linked together in networks in much the same manner as telephones. A computer may be a node on a network just like a telephone may be and indeed copper wires linking networked computers may be identical to the wires of a telephone network. In such a context the economic idea of efficiency arising from a network can be advanced to support rules that favor attachment—though whether potential efficiencies are achievable in a particular case will depend on the facts.[93]

---

[91] See, e.g., Peter S. Menell, An Analysis of the Scope of Copyright Protection for Application Programs, 41 Stan. L. Rev. 1046, 1066-1071 (1989); Office of Technology Assessment, Finding a Balance: Computer Software, Intellectual Property and the Challenge of Technological Change 197-199 (1992).

[92] Dennis W. Carlton and J. Mark Klamer, The Need for Coordination Among Firms, with Special Reference to Network Industries, 50 U. Chi. L. Rev. 446, 450 (1983).

[93] A broader definition of network externalities would include positive consumption externalities: “There are many products for which the utility that a user derives from consumption of the good increases with the number of other users who are in the same network.” Michael Katz and Carl Shapiro, Network Externalities, Competition, and Compatibility, 75 Am. Econ. Rev. 424 (1985). While this broader definition might

The problem with using the idea of network externalities in addressing the intellectual property protection of software is that the supposed externalities that are enumerated are usually not those having to do with a network in the node and link sense. Where advocates are seeking to justify replacement rather than attachment, the externalities are quite different. It is said, for example, that if all sellers of a particular kind of computer software—say, spreadsheet programs—adopted the same user interface, then buyers could shift from one company's product to another without the need to retrain employees.[94] This is indeed an externality—one that is analyzed at length below—but adding the word "network" in front of externality conveys little information. On the contrary, the arguments in favor of copying for replacement of application software (which is what is envisaged in the retraining argument) does not turn on whether the computers are networked; it applies, or fails to apply, equally to stand-alone computers.

A. Switching costs and lock-in

Even though the network externalities rubric is a misnomer where there is no network, it must be recognized that a firm with a highly successful original software product may have an advantage over even an innovating competitor where it is costly for users to switch. The question is whether the existence of switching costs should affect the availability of copyright protection.[95] To analyze that question, we must recognize that

---

encompass operating systems, where the user is likely to enjoy access to more application programs the more widely used the operating system is, it has much less relevance to application programs. The benefits to me of your using the same word processing program, for example, are much less important. For example, if I use Microsoft Word and you use WordPerfect, I can not only send you a letter but also send you an electronic file for you to edit, since I can save a Microsoft Word document as a WordPerfect file. Indeed, if I use a Macintosh, I can also save my document onto a dos formatted diskette so that you can edit it on your pc. In any case, this broader use of the adjective "network" has the disadvantage that it converts a large portion of the universe of economic decisions into "network" decisions, whether it is a case of buying a car, choosing a city to live in or even seeing a movie, since in each case one's utility may be affected by whether too few or too many make the same decision.

[94] See, e.g., Menell, supra note 91 at 1068.

[95] The literature on switching costs appears more relevant than the literature on network externalities simply because no network need be involved. But the switching cost literature comes to no clear conclusions about the impact of switching costs on competition. See, e.g., Richard Schmalensee, Product Differentiation Advantages of Pioneering Brands, 72 Am. Econ. Rev. 349 (1982); Joseph Farrell and Carl Shapiro, Dynamic Competition with Switching Costs, 19 RAND J. of Econ. 123 (1988); Benjamin Klein, Market Power in Antitrust: Economic Analysis after Kodak, 3 Sup. Ct. Econ. Rev.

switching costs are in no way special to software. Consider switching costs in everyday life. It takes time and even money to change checking accounts (going to the bank, buying new checks, etc.) Even more serious are the switching costs in buying a new or a used car (the time spent haggling with sales personnel, the sales tax that could be avoided by staying with one's old car, and so forth). Thus, switching costs are a regular feature of most markets for specialized, differentiated products.

The principal switching costs involved with application software involve training costs. Take the case of two application software products where there is no attachment issue (for example, where there are two independently derived application software products, both of which attach to a given operating system). The simple fact that is that there are likely to be training costs involved in buying a different application software product than one is accustomed to. These may be easily identified out-of-pocket costs, especially in the case of a firm changing software products already widely used by employees. Or they may be implicit costs generated by the temporary inefficiency attributable to learning by doing.

In considering the application of the switching cost argument to application software, it is important to note that the judicial decisions and the economic literature treating switching costs as a problem warranting special legal rules are concerned with the relationship between two products, not the simpler one-product case of switching bank accounts, automobiles or application software programs. In the judicial arena, for example, the Supreme Court decision in the *Kodak* case concerned the switching costs involved in Kodak's practice of making it difficult for users of Kodak copiers to switch from Kodak maintenance service to third party service by refusing to sell Kodak parts to third party service firms.[96] The copiers required ongoing service and at least some users wanted original Kodak parts used; hence there were really three products in Kodak: copiers, parts and service.

43, 49-63 (1993). Indeed, some of this literature suggests that switching costs can lead to lower prices. See Paul Klemperer, Price Wars Caused by Switching Costs, 56 Rev. of Econ. Stud. 405 (1989). In any case, this literature does not lead to the conclusion that the existence of switching costs should lead to the abolition of property rights.

[96] Eastman Kodak Co. v. Image Technical Services, Inc., 112 S. Ct. 2072 (1992). For a thorough treatment of the switching costs aspect of the case, see Klein, supra note 95. The legal issue involved whether there was an unlawful tie-in under the antitrust laws. No such issue arises, of course, where there is only one product, as in application software copyright infringement cases.

Similarly, in the *Data General* case, Data General refused to sell its operating system except in connect with Data General computers, thus making it costly for users who had purchased software written for the Data General operating system (and who therefore wanted to stay with the Data General operating system) to switch to computers from so-called "emulator" (clone) manufacturers.[97] Two products were "tied ": computers and operating system programs.

The *Data General* case illustrates why the switching cost argument is often referred to under the rubric of "lock-in." The central idea is that a user, having bought manufacturer *A*'s product *X*, which must be used over time with product *Y* (which the user has bought or continues to buy from *A*), will find it costly to switch to a new supplier of product *Y* if in order to do so he must also, for technological or contract reasons, buy a new product *X* from some manufacturer other than *A*. It is said the user is locked-in by his prior purchase of product *X* from manufacturer *A*.[98]

The economic literature on switching costs seems to have been stimulated by the development of the computer industry because, as in *Data General*, the lock-in effect was thought to be an inevitable aspect of the incompatibility between different hardware (and operating system) standards. As we have seen, the industry evolved through the development of clone machines to limit any such lock-in effect but even today some manufacturers, notably Apple, have been able to resist clones and to limit the choice of operating systems available for use on Apple machines.

[97] Digidine Corp. v. Data General Corp., 734 F.2d 1336 (9th Cir. 1984). The Data General case held the practice an unlawful tie-in, even though the market share of Data General in both the tied and tying market was very small. See the District Court opinion, In re Data General Corp. Antitrust Litigation, 529 F. Supp. 801 (N.D. Cal. 1981). In this respect the Data General precedent has not been followed by other circuits. See Will v. Comprehensive Accounting Corp., 776 F.2d 665 (7th Cir. 1985); and Frank H. Easterbrook, Intellectual Property is Still Property, 13 Harv. J. of L. & Pub. Pol'y 113 (1990).

[98] In speaking of Data General's customers being "locked-in," the Data General court was not speaking of users but that was due to the peculiarities of Data General's distribution system, in which it sold computers and operating systems to third parties who bundled them with compatible application software for resale to users. In Kodak the Supreme Court stated, "If the cost of switching is high, consumers who already have purchased the equipment, and are thus 'locked-in,' will tolerate some level of service-price increases before changing equipment brands. Under this scenario, a seller profitably could maintain competitive prices in the aftermarket if the switching costs were high relative to the increase in service prices, and the number of locked-in customers were high relative to the number of new purchasers." 112 S. Ct. at 2087.

B. Two product versus one product cases

Whatever the correct analysis of these two product switching cost problems, they are different from the one product case involving a user of an application program who, so long as he switches to a different application program written for the same operating system, does not have to incur costs associated with the necessity of also buying a new second associated product.

Any costs beyond the price of the new program are his own out-of-pocket costs—for example, retraining costs. As we have already seen, the economy is full of similar situations: witness the bank account and new automobile cases.

To be sure, there are some parallels between the one product and two product cases. One of the worries in the literature is that in the two product situation the seller of product *A*, having made the accompanying sale of compatible product *X*, may be inclined to act opportunistically, raising the price of product *X* because the user is now locked-in to *X* by his prior purchase of product *A*.[99] In the one product application program situation the seller could conceivably act opportunistically with respect to the price for upgrades or for additional copies of the program itself. But the ability to act opportunistically in this way is limited by the extent of the out-of-pocket costs involved in switching since there is no need to purchase a second new product in order to switch. Moreover, the user in the one-product case is well aware at the time of the initial purchase that he is locking himself in to this limited extent and that knowledge is itself a factor in the price that can be charged at the time of the initial purchase.[100] Indeed, in the world of firms acquiring many copies of an application program (which is the world envisaged in the training cost argument), the

[99] This was the Supreme Court’s worry in the Kodak case. See supra notes 95-96 and accompanying text. For a review of the economic literature, see Michael L. Katz and Carl Shapiro, Systems Competition and Network Effects, 8 J. of Econ. Persp. 93 (1994). This approach to tie-in sales ignores a standard economic explanation for tying products consumed over time to long-lasting products with which they are used, namely, price discrimination. See discussion and literature citations in Kenneth W. Dam, Fortner Enterprises v. United States Steel: “Neither a Borrower Nor a Lender Be,” 1969 Sup. Ct. Rev. 1, 15-17.

[100] This limited one-product opportunity for opportunistic behavior is present in all service industries. For example, one could as easily be worried about the opportunistic behavior of banks, who might impose higher bank charges once an account had been opened, of lawyers who might raise their hourly rate once work on a case began, and so forth.

seller's ability to act opportunistically is limited whenever a new sale is made.

C. Switching costs and incentives

The question with respect to switching costs in the application software context is, at base, not whether such costs exist but whether they are significant and, specifically, whether they outweigh the impact on incentives for innovators that would be involved in carving out an exception to copyright protection, especially an exception for what may be called "me-too copying"—that is, wholesale copying by a competitor who seeks to sell a competing, but substantially similar, product under his own trademark.[101] Most critics of software copyright would restrict such copying to the situation where the targeted program constituted a de facto standard. This makes sense from the standpoint of the new competitor who will find it hard to attract new customers to the extent that existing users of a de facto standard program would have to incur substantial training costs and to that extent be less likely to want to switch to the new product. On the other hand, the extent of training costs can vary greatly with the nature of the targeted program, being as easily high for programs with low market share and low for programs with high market share as vice-versa. To be sure, a competitor would be unlikely to want to copy a program with a low market share. Nonetheless, it is not possible to equate de facto standards and high training costs, and therefore the case for permitting copying of de facto standards products would have to turn on a factual inquiry in each case as to the competitive significance of training costs—an inquiry, lawyers might stress, for which a court would have no clear standard to apply.

One line of argument for permitting copying of any de facto standard, regardless of the extent of training costs, might be that software products rarely achieve de facto standard status. Indeed, according to this argument, that level of success occurs so rarely that, at the r&d stage, an application software firm would be unlikely to base its investment decision on the possibility that such a rare outcome would occur.[102] Such an argument would be based on a misconception. The r&d decision would depend on the expected return, which conceptually depends on the probability of each outcome times the profit associated with it. Though

[101] The me-too case is addressed at length, and distinguished from the case of a competitor who seeks to create an improved version while copying the copyrighted version as a base or kernel, infra notes 139-155 and accompanying text.

[102] Cf. Landes and Posner, supra note 47 at 352.

many software ventures have failed and mediocre results abound, major successes in application software, though infrequent, have led to large payoffs. For example, Lotus Development Corp. has become the third largest U.S. independent software company with a billion dollars per year in revenues largely on the basis of its 1-2-3 spreadsheet product; even today the latest version of that program remains its major product.[103]

The analogy to the importance of "giant fields" in the incentives for petroleum exploration is suggestive. In both industries there are literally thousands of firms motivated in part by the possibility, admittedly small, of a very large payoff. This motivation is rational; it simply reflects the principle that what counts in making an investment is the expected return, not the most likely return. A rule permitting copying of a product when it reaches de facto standard status must therefore have an effect on incentives. To return to the oil field analogy, a rule providing for the uncompensated confiscation of outsize discoveries would obviously have an impact on explo-

[103] Value Line Investment Survey 2118 (March 11, 1994). The very financial success of Lotus Development raises the question whether intellectual property protection can overcompensate. Acknowledging the argument about expected return made in the text, an issue is whether Lotus, for example, was overcompensated for the innovation in Lotus 1-2-3. One line of such criticism is that the Lotus menu tree was actually a minor innovation, analogous to the "*H*" automobile gearshift innovation referred to in Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003 (N.D. Tex. 1978), especially because Lotus was not the first firm to sell an electronic spreadsheet but rather built on the efforts of Visi-Calc (even though, as Judge Keeton found in the Lotus case, the crucial expression in Lotus 1-2-3was quite different from that in Visi-Calc. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 67) (D. Mass. 1970)). A second line of criticism, in contrast, would assume the importance of the Lotus innovation but nonetheless, viewing intellectual property as a monopoly, raise the question whether intellectual property protection is justified where it results in returns greatly exceeding the cost of innovation, a result that would in principle not be possible under competition. This second line of criticism, using the competition versus monopoly paradigm, would also argue that what counts in determining scope of protection is its cost and would lead to the conclusion that, as under competition, return should be related to cost. This issue arises with respect to all intellectual property rights in all industries. We do not seek, for example, to void patents simply because the return to the patentee seems excessive compared to his r&d costs. Quite the contrary, courts in some eras have positively favored essentially costless inventions under such rubrics as "flash of genius." In considering the overcompensation issue, the question should be whether copyright overcompensates software copyright owners as a class.

ration activity.[104]

In short, not only is the importance of switching costs a factual question in each case to which no a priori answer can be given, but it is also true that the denial of copyright protection whenever a software product achieves major marketplace success is bound to have an effect on the incentives for software r&d. Moreover, one result of such a rule, which would probably require litigation to determine its applicability in individual cases, would be to make intellectual property rights even more uncertain and poorly defined than they inevitably must be. Nonetheless, the idea of switching costs is a more precise and conceptually useful analytic construct for discussing the issues raised in the literature than are the concepts of compatibility, de facto standards and network externalities themselves. The reason is that the switching cost concept permits us to focus on the size of those switching costs rather than talking in conceptual generalities.

Finally, we can conclude that these three broader ideas (compatibility, de facto standards and network externalities) turn out to be not only closely related but essentially the same when used to condemn copyright protection. In all three cases the argument boils down to the proposition that the economy would be better off if follow-on firms could simply copy first-generation innovators' software products where it would be costly for existing users to switch to a new product. All that differs is the verbal justification and in fact all three arguments apply to roughly the same situation—where the first-generation firm has a leading position in the particular software market and a follow-on competitor seeks to market a replacement product. Compatibility is normally sought only with an industry leader's product; only a leader in a particular class of products can be thought to have created a *de facto* standard; and retraining "externalities" are quantitatively, in aggregate at least, most significant where the first-generation product is a market leader.[105]

---

[104] The discussion in the text abstracts from the question whether software firms or oil exploration firms are risk averse and whether they are able to diversify their risks. For a discussion in the context of oil exploration and the auction system of allocating exploration licenses, see Kenneth W. Dam, Oil Resources: Who Gets What How? 160-169 (1976). Denying copyright protection for highly successful software products would be analogous to the common governmental practice of changing tax and royalty rules after large oil fields are found, a practice that feeds back into the initial investment decisions of oil exploration companies. See id. at 173-179

[105] The text is concerned with the possibility of making an exception to copyright protection in the case where the first-generation firm's product becomes so successful that it can be called a de facto standard. The issue of copying of standards can arise in

VI. Contractual Solutions

Before pursuing in greater detail the case for and against copying to achieve replacement, we should examine more closely the use of contract to solve many of the problems that critics have perceived to be created by use of copyright protection. These contractual solutions are easiest to see in the attachment situation, and therefore they will be explored in that context before proceeding to consider their use in the replacement situation. We may accept the argument that in many cases there are gains to society from facilitating attachment of computer software programs, whether we think of them as gains from compatibility or standardization or from overcoming externalities. What is frequently overlooked is that first-generation firms have incentives to capture these gains. Let us take the case of an innovating firm that is first with a new kind of hardware or software product. The innovating firm may believe that it can gain from allowing, indeed encouraging, other firms to attach to its new product. If so, it has a variety of means to achieve that goal. Some are unilateral, others multilateral. Among the unilateral means are simply the communication of the necessary information to those who write complementary software for attachment. The form of the communication may be publication through manuals or written documents containing the requisite interface information.[106] But it may also take other forms, from working directly though informally with other software firms to more explicitly contractual forms such as licensing, joint ventures, and strategic alliances.[107] In short, firms may pursue standardization through contract.

other contexts. For example, an original innovator may simply fail to be successful, yet some aspects of his programs may be found in later products. These follow-on elements will usually be industry conventions that are unprotectable. See discussion infra, note 131-134 and accompanying text. This possibility is not fanciful. It is common knowledge that elements of the Microsoft user interface were created by Xerox and that Lotus 1-2-3 adopted elements of an earlier Visi-Calc program. Neither the Xerox nor the Visi-Calc programs survived commercially, though Lotus did buy Visi-Calc's rights. Clapes, supra note 54 at 42. These kinds of standards issues are not examined because the thrust of the inquiry here is to examine the economic arguments advanced to justify copying of successful programs.

[106] See, e.g., the discussion of ibm's Common User Access SAA Manual in Nimmer on Copyright §13.03[F][3] (1993).

[107] In Apple Computer, Inc. v. Microsoft Corp. 717 F. Supp. 1428, 1431 (N.D. Cal. 1989), the court observed, "Both Apple and Microsoft rely heavily on third-party programmers to develop application programs to run under their respective operating environments, thus enhancing the value of operating environments."

Indeed, if we use the language of externalities we may say that the firm can contract to overcome the externality.[108]

In many, probably in most, situations the original innovator has found it in its interest to try to enhance the number and quality of attaching products rather than trying to keep the market for attaching products to itself. Thus, computer manufacturers actively seek to increase the number of software firms writing programs for their particular hardware platform because they recognize that their success in doing so is likely to determine the market success of that platform. Similarly, firms selling operating systems usually try to increase the number and quality of application programs to run on their operating system.[109]

Firms may choose, however, to proceed in an overtly more multilateral way. Such approaches are commonly referred to as standardization. For example, firms in a particular industry may get together to create standards so that software programs may attach and work together in the same way that plugs and sockets permit machinery to work together. Sometimes these multilateral efforts are informal and ad hoc but often firms proceed under the auspices of a formal standard setting organization.[110] These multilateral efforts also involve at base the institution of contract and, to the extent one thinks of attachment as facilitating networking, they involve using contract to overcome network externalities.[111]

---

[108] This use of these various forms of contracting to deal with externalities is, of course, the essence of the Coase Theorem. See Ronald H. Coase, The Problem of Social Cost, 3 J.L. and Econ. 1 (1960). It is worth noting that in the computer software arena, the costs of transacting are relatively low because despite the total number of firms in the industry, the number involved in any particular software product area is much smaller.

[109] Katz and Shapiro discuss some of the alternatives under the rubric of “strategies to attract users to networks.” See supra note 99 at 103-105. The text focuses, for ease of illustration, on contractual initiatives taken, in the attachment situation, by an original innovating firm, say an operating system company. Alternatively, a follow-on firm with a new product—say an application program—may take the contractual initiative, seeking out the former firm in order to assure attachability.

[110] This kind of multilateral standard setting is described in Office of Technology Assessment, Global Standards: Building Blocks for the Future (1992), and much of the relevant literature is cited therein. See also National Research Council, Intellectual Property Issues in Software 70-71 (1991).

[111] One can, of course, postulate conditions under which the innovator might have incentives not to provide the requisite interface information to encourage other software firms to attach. Or an innovator might have incentives to try to prevent standards from being adopted that would facilitate attachment. One could postulate, for example, that the innovating firm might want to provide the attaching software program itself and

Although the use of contract as a solution to externality problems is most visible in the attachment world, it can equally well be used in the replacement world. Competitors can achieve gains by standardizing their products in order to create a larger market for their class of software or in order to share resources. Perhaps the most publicized recent example was the alliance between Apple and ibm to create a new operating system to compete with Microsoft.[112] The same kinds of initiatives are possible in the application software arena. We shall return to this theme in considering the improvement case, where one firm seeks to improve the application program of another but needs to copy the latter's product in order to be able to sell his new program successfully. All that is required is that there is a gain to be enjoyed, in which case the parties will have an incentive to share it through contract, provided that the follow-on firm is not enable to copy outright through an exception to the copyright law.[113]

VII. Competition among Standards

Before pursuing further the me-too and improvement cases, it is important to consider an alternative to permitting copying that is seldom discussed in the literature. That alternative is to encourage competition among standards by prohibiting copying even in de facto standard situations. Much of the economic literature assumes that competition among operating system programs or among application programs will not exist for long. This literature holds that such markets involving rival standards are "tippy" and therefore that one standard or the other is likely to triumph.[114] In other words, the reigning hypothesis concerning tipping is that the competition we observe in pc operating systems programs and in word processing and spreadsheet programs should not exist or at least is unlikely endure.

therefore might attempt to protect its market for the attaching product by withholding the information necessary for follow-on competitors to provide an attaching product. Or one might postulate that the innovating firm, believing that it had a superior market position, say through intellectual property protection or lower costs for its core product, might seek to "bundle" its core product with an attaching product so that the buyer who wanted the core product would be induced to acquire the attaching product from the innovator, not from a follow-on firm. These issues, which have formed the basis of public allegations against some leading firms, are beyond the scope of this article.

[112] See i.b.m. Now Apple's Main Ally, N. Y. Times D1 (Oct. 3, 1991).

[113] See discussion of the improvement case, infra notes 139-152 and accompanying text.

[114] See review of the economic literature in Katz and Shapiro, supra note 99 at 105-106.

Liebowitz and Margolis have pointed out that the tipping hypothesis, especially the notion that earlier inferior products are likely to exclude later superior products, is based largely on two historical incidents, vhs/Beta and qwerty/Dvorak, both of which have been badly misunderstood.[115] More recent work emphasizes that competition among standards does exist and that, in any case, there is no general principle that dictates that superior products will not displace earlier inferior products.[116]

If one looks to the software world today, competition among standards appears to be common, to persist and to lead to remarkable progress. Three illustrations from the pc world—operating systems, spreadsheets, and data base programs—reveal a common pattern. In operating systems Microsoft clearly has the lead with its ms-dos and its overlay, Windows, but it faces strong competition from a number of other operating systems. This competition continues even though Microsoft's ms-dos was first to the pc market. As Markoff observes:

Microsoft supplies the most popular operating system, ms-dos, controlling more than 80 percent of the pc operating system market. However, a number of competing programs are available, including System 7 from Apple Computer, Inc. for Macintosh computers, dr-dos from Novell Inc. and os/2 from [ibm].[117]

A broader definition of the market shows Microsoft accounting for only 66 percent of the operating system market in 1993.[118]

In spreadsheets the first successful "standard' was Visi-Calc, which was not only first to market but popularized the concept of the electronic spreadsheet. Yet it was displaced by Lotus 1-2-3, and today there are at least four major competitors: "Lotus's 1-2-3 and Microsoft's Excel are battling for lead position [while] Borland's Quattro Pro, and CA-SuperCalc, from Computer Associates, are also strong contenders."[119] In

[115] S.J. Liebowitz and Stephen E. Margolis, Network Externality: An Uncommon Tragedy, 8 J. of Econ. Persp. 133 (1994).

[116] Katz and Shapiro, supra note 99 at 108. On the value of product design competition in software, see Joseph Farrell, Standardization and Intellectual Property, 30 Jurimetrics 35, 48-49 (1989).

[117] John Markoff, Justice Department Considers Inquiry on Microsoft, New York Times 16 (Aug. 1, 1993). See Clapes supra note 54 at 23 (1993). dr-dos is now known as Novell dos. Lawrence M. Fisher, Microsoft's Operating System Rivals Get a Boost, Sort Of, New York Times, Sunday Business Section 7 (July 24, 1994).

[118] See graphic in John Markoff, Microsoft's Future Barely Limited, New York Times D1 (July 18, 1994).

[119] Mike Lewis, What's Between the Sheets?, Computer Weekly (Feb. 10, 1994).

1993 Lotus Development, far from having the de facto standard, trailed Microsoft 37.0 percent to 52.7 percent.[120] A similar story can be found in the data base field. Ashton-Tate was once the frontrunner, but when it failed to navigate successfully the transition from an Apple to a pc environment and to deliver upgrades on time, it quickly lost market position and was eventually acquired by another firm.[121] More recently its dBase product has faced strong competition from a variety of products with many new features and structures.[122]

The result of this competition among standards thus appears to be, if anything, an increase in the number of competing software products rather than the coalescence into one de facto standard product in each category that is assumed in some of the literature. In fact, with the rapid increase in power of microcomputers, they are increasingly coming into competition with so-called workstations which use a Unix-based operating system, and therefore a series of additional Unix-based application programs provide competition for more traditional microcomputer application programs.

The data on competiting standards call sharply into question the meaning of the familiar concept of a de facto standard. On the one hand, can a program be a de facto standard when it faces strong competition from competing standards? If so, can there be more than one de facto standard in a particular product category? If so, how many? If Apple's System 7 is number four in microcomputer operating system field, can it nevertheless be considered a de facto standard? If the answer is affirmative, on the ground that it is the dominant system for use on Apple microcomputers, can we consider Novell and ibm's operating systems, which trail Microsoft badly, also de facto standards? If we consider the relative losers in the competition among standards to be de facto standards, and therefore to be subject to copying by competitors, what are the implications for competition, not to speak of incentives for innovation?

Not only do standards thus compete and do those first to the market not always maintain their preeminence, but even more to the point from the standpoint of the user is the fact that the frontrunners at any one time can stay ahead only by continually upgrading their product. Microsoft

[120] See graphic in Markoff, supra note 118.

[121] Clapes, supra note 54 at 23.

[122] John L. Hawkins, The dBASE Report, Data Base Advisor (July 1990). Word processing programs show similar competition among standards. In 1993 Microsoft had 47.6 percent of the market and WordPerfect 35.7 percent. See graphic in Markoff, supra note 118.

may have been first to the pc marketplace through its initial contract with ibm but it stayed ahead of its rivals, including ibm itself with its os/2, by a number of major upgrades of its ms-dos and then by introducing an attractive new product, Windows, which though built on top of ms-dos, looks fundamentally different from ms-dos to the user and is one of the best selling software products in history. Today Microsoft is rapidly introducing a series of extensions and upgrades in order to be able to maintain its lead; for example, in a much publicized upgrade code-named Chicago, Microsoft will, according to its announcements, emancipate its operating system from its ms-dos platform.[123] In this light one can fairly ask whether, for example, consumers would have been better off if the original Microsoft dos program had been freely copiable on the ground that it had become a de facto standard and therefore it was important to save on training costs. No doubt in such an environment ms-dos would have survived and Microsoft would still have been forced to innovate to a certain extent in order to stay abreast of increases in computer power, but it is hard to conclude that users would have been better off if ms-dos had been, in effect, expropriated, all in the name of compatibility, de facto standards or network externalities. What would have been lost was user interface competition that has created not only alternatives to the Microsoft standard but also more competitive pressure on Microsoft itself to innovate. Thus, competition among standards, with the resulting impetus for rapid improvements in leading products, is arguably much more important for users than any reduction in prices that might result from allowing copying of those leading products.[124]

---

[123] See Christine Burns, Microsoft Making Travel Plans for its Operating Systems: Vendor Stopping in Daytona, Chicago and Cairo, Network World (Dec. 17, 1993/Jan. 3, 1994); Ed Bott, Inside Windows 4.0; Microsoft Corp.'s Chicago Operating System, 7 pc Computing 124 (March 1994).

[124] User interface competition in application software has another important advantage that is often overlooked in the literature. Such competition has largely avoided the problem, so often discussed, of premature coalescence around de facto standards—otherwise sometimes referred to as the qwerty keyboard problem. See, e.g., Joseph Farrell and Garth Saloner, Installed Base and Compatibility: Innovation, Product Preannouncements, and Predation, 76 Am. Econ. Rev. 940 (1986). Postulating that more efficient keyboards than the familiar qwerty keyboard exist, Farrell and Saloner explain that premature convergence on the qwerty keyboard created “excess inertia” making transition to a better standard infeasible. Liebowitz and Margolis have argued with copious evidence that the oft-repeated story of the inferiority of the qwerty keyboard relative to the later Dvorak keyboard is a myth. S.J. Liebowitz and Stephen E. Margolis, The Fable of the Keys, 33 J. of L. & Econ. 1 (1990).

VIII. The Me-too, Follow-on Product Case

The fact that standards competition not only exists and that later superior products not only succeed but sometimes displace earlier “de facto standards” is especially important in the contemporary litigation context because the striking fact about the software copyright cases is not just that nearly all involve replacement rather than attachment. They often also involve a struggle between one firm that, as innovator, was first to a new market with a popular software program and a second firm that, coming along later, sought to sell a competing product that was as close as possible, from the buyer’s perspective, to the original innovating product.[125] This me-too copying is to be distinguished from the case, less common in the judicial decisions, of a struggle between an initial innovator and a follow-on innovator who tries to compete by putting a new and better product based on the original product into the same marketplace. This latter case we may call the substantial improvement case.[126] Indeed, it is because the bulk of the cases involve a me-too product that they usually deal with the user interface, especially screen displays. For these reasons, the economic arguments that are used in attacking copyright protection warrant being tested in the replacement context and, specifically, the me-too case.[127]

[125] See Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1244 (3d Cir. 1983); Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1228-1229 (3d Cir. 1986); Computer Associates Int, Inc. v. Altai, Inc., 982 F.2d 693, 698-700, 718-719 (2d Cir. 1992). For example, in Lotus Dev. Corp. v. Paperback Software Int’l, 740 F. Supp. 37, 69-70 (D. Mass. 1990), the follow-on firm not only copied but was able therefore to “truthfully declare” in its VP-Planner manual:

> VP-Planner is designed to work like Lotus 1-2-3, keystroke for keystroke....VP-Planner’s worksheet is a feature-for-feature workalike for 1-2-3. It does macros. It has the same common tree. It allows the same kind of calculations, the same kind of numerical information. Everything 1-2-3 does, VP-Planner does.

Although there were differences between the Lotus and Paperback user interfaces, Judge Keeton was much more impressed with the similarities: “From the perspective of both an expert and an ordinary viewer, the similarities overwhelm differences.” 740 F. Supp. at 70.

[126] The substantial improvement case is discussed infra notes 139-152 and accompanying text.

[127] The de facto standard argument for an exception to copyright protection applies primarily to replacement rather than attachment. The question of the protection of interfaces in the attachment context raises a more complicated set of questions. In any event, the interfaces involved in that context are usually not user interfaces but rather internal interfaces.

In the me-too case, the argument for permitting copying is essentially a conventional competition argument rather than an intellectual property argument. Permitting me-too copying is not necessary to permit future innovation because, by definition, the me-too copier does not seek to innovate.[128] Moreover, permitting me-too copying restricts the incentives for the first generation firm to innovate. For these reasons, some courts have recognized the inconsistency between the de facto standard argument and intellectual property principles and the irony that would be involved in protecting minor innovations, while denying protection to major innovations that gain wide consumer acceptance, by rejecting any "commercial necessity" argument for copying.[129] Although some analysts would prefer to incorporate conventional competition arguments in the interpretation of intellectual property laws, the result might well be to undermine the coherence of both fields of law since the antitrust laws remain available to attack true market monopolies and restraints of trade in software markets. Indeed, the de facto standard argument has been advanced primarily to support me-too copying where market power in the antitrust sense did not exist. Thus, to deny protection on competitive grounds is to introduce a kind of mini-antitrust law for intellectual property cases that we would be unwilling to apply in other areas of law.[130]

---

[128] In the improvement case, the de facto standard argument has more relevance. See infra notes 139-152 and accompanying text.

[129] This was a key point for Judge Keeton in the Lotus litigation: "By arguing that 1-2-3 was so innovative that it occupied the field and set a *de facto*_standard, and that, therefore, defendants were free to copy plaintiff's expression, defendants have flipped copyright on its head. Copyright protection would be perverse if it only protected mundane increments while leaving unprotected as part of the public domain those advancements that are more strikingly innovative." Lotus Dev. Corp. v. Paperback Software Int'l, 740 F.Supp. 37, 79 (D. Mass. 1990). See also Allen-Myland v. International Business Machines Corp., 746 F. Supp. 520, 533 (E.D. Pa. 1990): "Otherwise, a computer program so complex that vast expenditures of time and money would be required to develop a different program expressing the same idea would not be protected, even if innumerable different programs expressing that idea could be written, while a simpler program requiring less significant expenditures of time and money might be protected." See also Consul Tec, Inc. v. Interface Systems, Inc., 3 CCH Computer Cases ¶ 46,685 (E.D. Mich. 1991).

[130] The competition argument for a de facto standards exception assumes that the follow-on firm is at an unnatural disadvantage if prevented from copying. It is often overlooked that even a follow-on firm that elects not to copy nonetheless has cost advantages over the original innovating firm. For example, the follow-on firm is able to determine from the success and failure of earlier innovators what works and what does not and what appeals to users and what does not. Hence, even without copying, the

In considering the de facto standard argument in favor of copying to facilitate me-too products, it is a sign of the weakness of the argument that it is often interwoven with a number of software-specific arguments. For example, it is asserted that copyright owners should not be permitted to protect aspects of their programs that are so popular that they have become part of the common repertory or tool kit of software writers. The short answer is that software copyright doctrine already denies protection for unoriginal aspects of an otherwise original target program. In doing so, copyright law simply applies well-established principle from non-software cases that limit protection to original expression. For example, a target program receives no protection for aspects of that program that are in the public domain[131] or that were copied from predecessor programs.[132] Similarly, one cannot copyright an industry convention,[133] just as one cannot copyright a word in the English language or a popular expression.[134] To that extent the de facto standard argument adds nothing

follow-on firm benefits from the efforts of those who went before. Cf. Easterbrook, supra note 99at 115. Moreover, the notion that copyright primarily protects large firms against smaller rivals is, at best, an exaggeration. For example, Lotus, founded in 1982, was small when it first launched Lotus 1-2-3, perhaps smaller than Software Arts, the owner of Visi-Calc. See description in Clapes, supra note 54 at 42. Moreover, even de facto standards do not always belong to large firms. An illustration from the field of software-related patents involves Stac's compression technology, which was almost certainly a de facto standard but, according to the jury, was improperly used by Microsoft. See infra note 174.

[131] Computer Associates Int'l v. Altai, 982 F.2d 693, 710 (2d Cir. 1992).

[132] Apple Computer v. Microsoft Corp., 779 F. Supp. 133, 134-135 (N.D. Cal. 1991).

[133] See Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1023-1024 (N.D. Cal. 1992), refusing protection where features are incorporated in most graphical user interfaces. Principles permitting copying of conventional terms, definitions and structures are not peculiar to software but rather apply equally to written works. See Business Trends Analysts v. Freedonia Group, 650 F. Supp. 1452, 1461 (S.D.N.Y. 1987); McGraw Hill, Inc. v. Worth Publishers, Inc., 335 F. Supp. 415, 421 (S.D.N.Y. 1971) (copyright does not protect economic terminology in an economics textbook): "The long history of the development of the discipline has resulted in a large corpus of technical words and phrases whose meaning have been fixed, and of concepts whose importance is generally acknowledged. No professor dealing in the basics could hope to be successful with a radical departure from either the jargon or the substantive professional consensus, on issues where one exists." In all of these examples, the conventions in question are, in effect, in the public domain.

[134] The fact that individual words in the English language cannot be copyrighted does not mean that a passage composed of those words cannot be copyrighted; Nimmer gives the illustration of Hamlet's soliloquy, where a Shakespeare writing today could not protect "to," "be," "or," and the like but could surely protect the soliloquy. Nimmer on Copyright § 13.03[F] n. 345.1 (1993). An original compilation is copyrightable, even

to existing copyright doctrine. To be useful, the de facto standard argument must apply to the original aspects of the target program.

Perhaps precisely because the de facto standard argument is weak in the me-too replacement context, the notion of network externalities has been piled on top of that argument in order to bolster it, even though, as previously discussed, the de facto standard and network externalities arguments are at base one and the same.[135] Nevertheless, assuming that major externalities are involved, the economic question is whether the elimination of property rights in highly successful user interfaces is the best approach. As the voluminous literature on the Coase theorem, and indeed the original Coase article, make clear,[136] contract is a superior alternative where transactions costs do not preclude bargaining, including situations where copyright has customarily been used.[137] We have already seen that the institution of contract has an impressive record in the attachment product world where hundreds, often thousands, of application software firms are involved.[138]

In the replacement product world, the number of firms involved with respect to a particular class of products is necessarily very much smaller and therefore there is no substantial barrier to bargaining. If the prior innovating firm believes that costs to users (such as training costs) would fall by promoting identical user interfaces among competing products, then there are no obvious barriers to concluding the appropriate contractual arrangements either directly with competing firms or through the standardization process. If the cost savings to users are great enough, the innovative firm would have an incentive to capture some of them by an inclusive (sometimes called "open ") strategy. Critics might object that the innovative firm would gain a disproportionate share of whatever financial benefits accrue to the software industry from such a form of

though the individual elements of the compilation are not. See the definition of "compilation" in 17 U.S.C. § 101 ("a work formed by the collection and assembly of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the work as a whole constitutes an original work of authorship"); Feist Publications, Inc. v. Rural Telephone Service Co., Inc., 499 U.S. 340 (1991): Kregos v. Associated Press, 937 F.2d 700 (2d Cir. 1991); and Atari Games Corp. v. Oman, 888 F.2d 878 (D.C. Cir. 1989). Similarly, even though common geometric shapes may not be copyrightable, their arrangement may be. See North Coast Industries v. Jason Maxwell, 972 F.2d 1031, 1035 (9th Cir. 1992).

[135] See discussion supra notes 86-105 and accompanying text.

[136] Coase, supra note 198.

[137] Landes, supra note 47 at 105-106.

[138] See discussion supra note 106-113 and accompanying text.

standardization. While that might be true (and if so would be a reward to innovation), the distribution of profits would in the normal case have no effect on the achievement of the resulting cost reduction to users and to the economy as a whole.

IX. Beyond Me-too Copying

Although me-too copying cannot be defended on economic grounds, more complicated economic issues arise in two other cases: copying a competitor's application program as a base or kernel for an improved product (previously referred to as the improvement case), and intermediate copying of an operating system interface in order to write a new application program.

Both the me-too and the improvement cases involve one of the fundamental distinctions in copyright law. The Supreme Court in 1994 dealt briefly in the *Campbell* case with a distinction between a "transformative" and a "substitutive" use.[139] The Court contrasted "commercial use [that] amounts to mere duplication of the entirety of an original" and "serves as a market replacement for it" with use that is "*transformative*."[140] Justice Kennedy's concurring opinion called the former use "*substitutive*."[141] Earlier, in the *Sony* case, the Supreme Court had distinguished between a "productive" and a "unproductive" use.[142] This distinction, whatever the exact words chosen, is well established in the law and economics literature,[143] and is implicit in the fair use provisions of Section 107 of the Copyright Act, which make "the purpose and character of the use" and "the effect of the use upon the potential market for and value of the copyrighted work" two of the four statutory fair use factors.

Under the *Campbell* test, me-too copying is a substitutive use; its purpose is to replace an existing product with a substantially similar

[139] Campbell v. Acuff-Rose Music, Inc., 114 S. Ct. 1164 (1994).

[140] Id. at 1171. (Emphasis supplied)

[141] Id. at 1180, 1181 (Emphasis supplied)

[142] Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 455 n. 40 (1983) (emphasis supplied) and id., dissenting opinion, 457, 478-479.

[143] See Landes and Posner, supra note 47 at 360, distinguishing between a "productive" and a "reproductive" use. ("A productive use is one that lowers the cost of expression and tends to increase the number of works, while a reproductive one simply increases the number of 'copies' of a given work, reduces the gross profits of the author, and reduces the incentives to create works.") And see the application of this distinction in Richard A. Posner, When Is Parody Fair Use?, 21 J. of Legal Stud. 67 (1992). See the use of the term "transformative" in Pierre N. Leval, Toward a Fair Use Standard, 103 Harv. L. Rev. 1105, 1111 (1990).

product. But suppose that the purpose of copying is to bring a better application program product to the market by using the existing product as a base or kernel on which to build enhanced or additional features. Obviously a rule treating such a use as transformative and hence allowing the copying could lead to abuse, and therefore one has to assume for purposes of analysis that the enhanced or additional features are of major importance and not merely a subterfuge for permitting me-too copying. More specifically, one has to assume that the value added by the improvement constitutes at least a major portion of the value of the new product because otherwise too pat a formula for avoiding the rule against me-too copying would be created. But if the improvement is truly important and creates a major portion of the value of the new product, then the issue is squarely posed. Should this form of copying be treated as copyright infringement or as fair use? Note that here the attachment/replacement distinction is not particularly helpful because, while the purpose is to replace the copied product, the situation bears some semblance of attachment in the sense that the follow-on firm adds something of value from the standpoint of users.

In such circumstances the fourth statutory fair use factor concerning the effect “upon the potential market for and value of the copyrighted work” surely argues for the use being infringing. On the other hand, failure to allow copying might have the cost of unduly hampering future innovation, depending on whether the copying was necessary to bring a new and better product to the market. To that extent the use is transformative under the first fair use factor involving “purpose and character of the use.” Perhaps it will be a relatively rare case in which there were no physical alternatives to copying; normally the enhanced or new features could have been added to a new base or kernel. The analogy to the rule implementing the expression/idea dichotomy (concerning whether there is one or several ways of implementing an idea) is suggestive of a possible rule. In any case, the argument for permitting copying is unlikely to be that copying is physically necessary but rather that it is necessary in order to be able to attract existing users to the new product.[144] Thus, the issue is whether wholesale copying of an existing application program where necessary from a marketing point of view to

[144] If what is being copied can be said to be public domain industry conventions, as opposed to a program as a whole or program elements that are quantitatively or qualitatively significant, then copying is already permitted. See the discussion of industry conventions, supra notes 133-134 and accompanying text.

bring a substantially improved application program successfully to market should be permitted.[145]

Ultimately the economic choice for copyright law in the case just postulated boils down to the effect on incentives for innovation versus the effect on future innovation. Since the new replacement product competes with the old and may in fact displace it from the market, the negative effect on incentives for innovation of a rule permitting copying is clear. On the other hand, the beneficial effect of such a rule on future innovation is equally clear.

Even if one were to carve out an exception to copyright protection in de facto standard cases in order to encourage future innovation, a court would still have to look at the facts of the particular case. Specifically, the court would have to decide how much innovation would be required in the follow-on product relative to the copied features in order to justify an exception to copyright protection. If one adopts the view articulated above that we should look solely to intellectual property principles, seeking to balance present and future innovation (leaving competition issues to the antitrust laws),[146] then minor improvements in the follow-on product should not be enough and therefore the court would have to inquire as to that balance in resolving a claim of infringement. In contrast, Judge Keeton, in both his *Paperback* and *Borland* decisions, refused to look at that question where significant features of the copyrighted program were copied.[147]

Even if this approach might lead one to permit copying in the postulated circumstances of major improvements to a de facto standard program, two important considerations point in the opposite direction. The first consideration derives from the notion of competition among

[145] In principle, the same issue could arise in the case of the copying of an operating system program in order to offer an improved operating system program. Here, however, the criterion of the improvement being of major importance in the sense of constituting at least a major portion of the value of the new product is unlikely to be met because of the complexity of operating system programs. Hence, copying of operating system programs appears likely to constitute a substitutive use.

With regard to either operating system or application programs one can make the legal argument that even a transformative upgrade violates the copyright owner's exclusive right under § 106 to prepare derivative works, which in the case of software presumably includes upgrades (e.g., version 3.0 for version 2.0).

[146] See discussion supra note 130and accompanying text.

[147] Lotus Development Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 70 (D. Mass. 1990); Lotus Development Corp. v. Borland Int'l, Inc., 788 F. Supp. 78, 97-98 and 799 F. Supp. 203, 220-222 (D. Mass. 1992).

standards. The tendency of a rule permitting copying would be to drive the entire marketplace for a particular kind of application program toward coalescence around the standard that first became popular in the marketplace. It would increase any "tipping" tendency of standards and indeed might do so prematurely.[148] A rule prohibiting copying, in contrast, is more likely to lead to competing standards in the marketplace. As previously argued, competition among standards has worked and may well have the effect of inducing even more rapid progress in performance and technical innovation.[149]

The second consideration is perhaps more subtle. The effect of a rule calling on the courts in each particular case to determine whether copying of an application program was, under the *Campbell* case, transformative or substitutive would greatly reduce certainty as to intellectual property rights. In many cases it would require litigation to make the determination.[150] The consequences of uncertainty in property rights are manifold. But one consequence warrants particular attention.

When property rights are clear, a property rule of remedies can produce important benefits. If the original innovator can enjoin the copying, then the property rights are clear and the parties will have an incentive to deal with each other through contract. The follow-on competitor will have an obvious incentive to seek a license from the copyright owner. Less obviously, but perhaps even more important, the original innovator will have an incentive to contract with the follow-on competitor if the improvement is truly important. If incorporating the improvement would lead to increased sales (including perhaps higher prices), then there is a gain that both parties will have an incentive to realize through contract.[151]

One can object that because there are, by hypothesis, only two parties to the negotiation, a type of bilateral monopoly is present, strategic bargaining behavior is likely over how the gain is to be shared, and in the end no agreement may be reached. Still, the software world is replete with

---

[148] It would thus exacerbate any tendency toward tipping and toward premature coalescence on de facto standards that some have purported to find. See supra notes 114-116 and accompanying text and note 124.

[149] See discussion of competition among standards supra notes 106-113 and accompanying text.

[150] D.C. Toedt, Oh, Pretty Woman: Muddying Software Copyright Even Further with 'Transformative Fair Use,' Computer Lawyer 15, Vol. 11, No. 6 (June 1994).

[151] See discussion of contractual solutions supra note 106-113 and accompanying text.

such agreements. Many forms of such agreement fill the newspapers and trade press: licenses, joint ventures, strategic alliances and even acquisitions. The important point is that the negotiation of such agreements, good for parties and users alike, is simpler and more likely to achieve success where property rights are clear. Moreover, the nature of the parties who are negotiating, being business firms all involved in writing software, means that transactions costs are likely to be low.[152]

A second form of copying beyond the me-too situation warranting analysis, but that is beyond the scope of this article, is intermediate copying of an operating system interface (and, by analogy, copying of an application program interface) in order to write a wholly new application program. Unlike the case of copying of a competing product to launch an improved version where the attachment-replacement distinction is clouded, this is clearly a case of attachment. By hypothesis, the new application program does not compete with the operating system whose interface is the subject of the intermediate copying. Moreover, the balance in the *Campbell* transformative-substitutive test shifts sharply toward the transformative side.

This form of copying normally involves what is known as the decompilation, or disassembly, form of reverse engineering. The notion is that what is involved is intermediate copying only in order to be able to write independent programming so that the new application product will attach to the operating system. The assumption is that the final version of the new product will not include any of the protected expression in the copied program. The intermediate copying is, by hypothesis, required to decompile, or disassemble, object code in order to be able to obtain source code so that the application program firm will be able to write its own program to permit attachment.

How often such decompilation is necessary from the standpoint of application software firms is not clear because operating system firms have an incentive to make the requisite interface information available,

[152] This is not a situation where the parties are dispersed, as in cases involving consumers or in the oft-cited environmental harm situation, where high transactions costs make contracting less efficient. Moreover, although the case of the lone software writer who is able single-handedly to improve major software programs commands romantic appeal, economic incentives lead most software engineers to collaborate in firms. Hence the universe of potential contracting parties is limited and knowable and therefore transactions costs are relatively low. Finally, the problem that may have been involved in the Campbell case, namely, that the owner of the song copyright may have refused on principle (or perhaps out of prejudice) to negotiate with the black rap artists who wanted to produce the parody, is less likely to be presented in the software context.

either directly or by manuals or in other ways, in order to increase the number of application programs that run on its operating system. However, in connection with situations like *Sega/Accolade*, a video game case,[153] some have argued that some intermediate copying may nonetheless be necessary to facilitate attachment.

A general exception for the decompilation form of reverse engineering could present a serious backdoor opening for outright wholesale copying. The issue therefore received considerable attention in the drafting of the European Union's software directive where a Council Directive permitted it in circumstances where "indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs," but imposed, among several restrictions, the prohibition of any decompilation for the purpose of developing a "substantially similar" program.[154] While the issue of intermediate copying is beyond the scope of this paper, it is clear that the twin distinctions between attachment and replacement and between transformative and substitutive uses provide key concepts for the analysis of particular factual situations.[155]

X. Software-Related Patents

Software may benefit from patent was well as copyright protection.[156] However, patent protection differs not just in theory but in scope and content from copyright protection. A 1992 advisory commission concluded that the two forms of protection "cover different aspects of an

---

[153] Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992). See also Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832 (Fed. Cir. 1992).

[154] Council Directive 91/250, 1991 O.J. (L 122) 442, Article 6(1) and 6(2)(c).

[155] The issue of intermediate copying can also arise in the case, discussed earlier in the text, of the follow-on application program firm that seeks to develop an improvement for a competitor's existing application program. In order to determine what part of the existing program to include in its own follow-on program, it must first understand that program, which may lead it to decompile object code. This also raises the question, sometimes discussed, of whether it is permissible to engage in intermediate copying in order to understand and use the ideas (as opposed to the expression) in a program. For a skeptical view of the need for such intermediate copying in view of alternative means of learning those ideas, see Miller, supra note 5 at 1015 n. 182.

[156] For a summary of the history leading to patent protection for software, see Peter B. Maggs, John T. Soma and James A. Sprowl, Computer Law 185-186 (1992). See also Diamond v. Diehr, 450 U.S. 175 (1981), including the dissent by Justice Stevens at 193.

article, and thus do not conflict with each other."[157] Although this statement should not be read to imply that there is no overlap whatever in what is protected, it is certainly true that patents afford protection in a completely different way.[158]

Copyright protects expression, which means both software code and certain non-literal elements, whereas patents protect invention. Copyright does not protect ideas. Though it is sometimes assumed in some copyright literature that patents do protect ideas, this is true only in a quite loose and general way. For example, patents do not protect abstract ideas; rather, they protect ideas only when a practical manifestation for those ideas has been shown by the patent applicant.[159] Moreover, patents protect only novel, useful and nonobvious ideas.[160] Finally, the Supreme Court has said that mathematical algorithms are may not be patented, a qualification that has significance for patent protection in the case of software.[161] It is consequently generally believed that computer software can qualify for patent protection only to the extent that the software interacts with or acts upon the hardware of a computer or some other

[157] Advisory Commission of Patent Law Reform, A Report to the Secretary of Commerce 146 (Aug. 1992) (hereafter "Advisory Commission Report"). See Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832, 839 (Fed. Cir. 1992)

[158] The assumptions of the two systems as to their purposes differ. Goldstein observes that copyright strives for "the widest possible differentiation...by protecting all original expressions, however little they differ from earlier works, as long as they do not infringe earlier works." He states that patent law "reflects the belief that society's interest in technological advance will best be served by granting protection only to inventions that represent substantial advances over the prior art." Hence, "copyright law sets a low standard for protection and attaches a correspondingly thin array of rights to the works protected" whereas patent law "set[s] the standards and level of protection high...." Paul Goldstein, Copyright § 2.15 (1989). On the relation between copyright and patent protection for software, see Joint Overlap Study 48-89.

[159] The utility requirement, 35 U.S.C. § 101, together with the enablement and best mode requirements, 35 U.S.C. § 112, assure that patents are not granted for abstract ideas.

[160] 35 U.S.C. §§ 101-102.

[161] Diamond v. Diehr, 450 U.S. 1 (1981). See Donald S. Chisum, Patents § 1.03[6] (1993). The pto has taken the position that only *mathematical* algorithms are excepted from patentability "since any process is an 'algorithm' in the sense that it is a step-by-step procedure to arrive at a given result." Patentable Subject Matter, Mathematical Algorithms and Computer Programs, 1106 Official Gazette of the Patent and Trademark Office 5 (Sept. 5, 1989), reprinted in Maggs, Soma and Sprowl, supra note 156 at 264, 266. Anecdotal evidence suggests that the significance of the mathematical algorithm exclusion is declining, and indeed the reason for the exclusion is not clear if the other criteria for patentability are met, namely, that the algorithm is novel, useful and nonobvious and is not an abstract idea.

apparatus. As the 1992 advisory commission stated, "[A]though computer program-related inventions use mathematics or can be understood in terms of mathematics...,patents for computer-related inventions are directed to performing commercial and technological methods or processes or to apparatus using programs to control the operation of hardware."[162] These considerations explain why patents involving software are often referred to as software-*related* patents and why most software claims today are drafted as "apparatus claims" so that it is the program's effect on the "apparatus" (which may be a computer or part of a computer), rather than the code itself, that is being claimed.[163]

The hardware-related aspect of software-related patents also follows in part, though not necessarily inescapably, from the legal requirement that utility patents can be granted only on a "process, machine, manufacture, or composition of matter."[164] A software-related patent therefore normally covers either a "process" (which would normally imply some apparatus upon which the process operates) or a "machine" as it is governed by software. In that sense one could say that it is not the software code as such that is patented or, alternatively, one could say that one cannot obtain a patent on software standing alone.

Despite these limitations a computer software firm setting out today to write new software must take into consideration the possibility that it will be limited in what it can do by both patents and copyrights belonging to a prior innovator. But there is a crucial difference between patents and copyrights. Since independent creation is a complete defense to copyright infringement actions, a firm does not have to worry about copyrights belonging to others so long as it does its own work.[165] For patents,

[162] See John P. Sumer, The Copyright/Patent Interface: Patent Protection for the Structure of Program Code, 30 Jurimetrics 107, 112 (1989).

[163] See In re Iwahashi, 888 F.2d 1370 (Fed. Cir. 1989); Notice Interpreting In re Iwahashi (Fed. Cir. 1989), 1112 Official Gazette of the United States Patent and Trademark Office 16 (March 13, 1990), reprinted in Maggs, Soma and Sprowl, supra note 156 at 280. See also Arrhythmia Research Technology, Inc. v. Corzonix Corp., 958 F.2d 1053 (Fed. Cir. 1992). As Clapes observes, "What salvaged [the] patent in [Diamond v. Diehr] was the fact that the process claimed...was not a mathematical process but an industrial process: the application of computers to curing rubber in a particular way. The significance of that distinction has not be lost on patent lawyers. Thousands of software-related patents have been filed.... all based on methods of accomplishing a real-world result by using a computer under the control of a program." Clapes, supra note 54 at 32.

[164] 35 U.S.C. § 101.

[165] Two practical caveats to the statement in the text: *(1)* there is anecdotal evidence that employees under pressure from management may copy to meet deadlines; and *(2)*

independent creation is not a defense and therefore a patent search is prudent, at least before commencing marketing.

In principle, the economic issues involved in software-related patents raise no economic issues other than those presented by patents generally.[166] However, the actual administration of the system for examining software-related patent applications has defects that are generally acknowledged and that do raise the possibility that software-related patents unduly favor present innovation at the expense of future innovation. The possibility of a departure from an optimal balance between innovation today and innovation tomorrow arises from the failure of the Patent and Trademark Office (pto) to respond adequately to the explosive growth in software and to the high degree of innovation in the industry, particularly in its early years. Since patent applications are normally unopposed and the proceedings are secret, the effectiveness of the examining process in assuring that the requirements of novelty and nonobviousness are met depends crucially not just on the skill of the examiner but also on the quality and accessibility of the files available to the examiner. For example, an examiner can normally determine that an application fails to meet these criteria only by finding pertinent prior art in the pto files.[167] The 1992 Advisory Commission particularly criticized the unavailability and inaccessibility of non-patent technical software

since access plus substantial similarity is sufficient to prove copyright infringement, Gaste v. Kaiserman, 863 F.2d 1066 (2d Cir. 1988), it is possible that a new program may engender an infringement action where an employee has had access, even though that employee intended to do his own work. Moreover, for popular mass-market programs, the risk is that access may be presumed. See discussion supra note 61. See also the concept of subconscious infringement in ABKCO Music, Inc. v. Harrisongs Music, Inc., 722 F.2d 988 (2d Cir. 1983).

[166] Important economic aspects of patent law are analyzed in Dam, Economic Underpinnings.

[167] These files are generally regarded as inadequate with respect to prior art other than prior patents. To be sure, the patent applicant is required to draw prior art of which he is aware to the attention of the pto, 37 C.F.R. 1.56, but since there is a good deal of reinventing of the software wheel, this requirement may be less effective than in other fields. These and similar concerns were ventilated at length in software patent hearings held by the Patent and Trademark Office in 1994. Among the factors cited in the Notice of Hearings as contributing to difficulties in uncovering prior art were *(1)* “Early programming techniques were not well documented or publicly available,” *(2)* “Locating and obtaining the most relevant prior art is extremely difficult, due to the widely diverse nature of processes that have been implemented by computer software-related systems;” and *(3)* “Software is not documented in a consistent, readily understandable format....” 58 Fed. Reg. 66347, 66351 (Dec. 20, 1993).

literature of the kind that might reveal prior art. It equally criticized inadequacies in classification and indexing of both patent and non-patent software literature. The Commission made a series of recommendations bearing on these issues as well as on recruitment and training of examiners in the software area.[168]

Although it may be true that in many industries patents are frequently issued that turn out, in subsequent litigation, to be invalid for lack of novelty or for obviousness, the fact that this problem appears to be much more severe with respect to software-related patents creates a special problem of deterring software innovation. Even though in non-software industries the prospect of having to resort to litigation to launch a product covered by a dubious patent is a deterrent, software has several characteristics that could make the deterrent more severe if the incidence of invalid software-related patents is especially high. One is that the software industry, despite its increasing maturity, still involves thousands of firms, many quite small,[169] and therefore the feasibility of litigation for many follow-on competitors is problematical. Second, many software programs are sufficiently complex that they may potentially raise infringement issues under many patents.[170]

For these reasons many smaller software firms have criticized the institution of software-related patents as creating special risks not present, or at least not so common, in other industries, such as "not being able to find the patents which apply to a program being developed; not being able to determine if the patent applies to a particular program [and] not being able to find prior art (e.g., evidence that a technique had been previously used although not published)."[171] In late 1993 the

---

[168] Advisory Commission Report 33-34.

[169] Id. at 145.

[170] See statement of Daniel Brickman, in Robert Patrick Merges, Patent Law and Policy 88 (1992), and statement of Jerry Baker, Public Hearing of Use of the Patent System to Protect Software-Related Inventions (pto, Jan. 26, 1994) ("pto Software Hearings"). The protracted length of some patent litigation may produce a further problem, especially in view of the relatively short live of some software products.

[171] Advisory Commission 147. The quotation is a summary by the Commission of complaints made in submissions to it. Another complaint was that certainty could not be achieved without litigation that was "costly in both time and money." Ibid.

The familiar concern of economists about the length of patent protection, see Dam, Economic Underpinnings at 257-259, might be thought to play a special role with respect to software, in view of the rapid evolution of the state of the software art. While patent doctrine plays an important role in determining the breadth of patent protection, it plays no role with respect to length since the term is fixed by statute. Ibid. It is worth recalling that many contemporary software "innovations" were actually invented decades ago but announcement of the issuance of a multimedia search and retrieval patent gave rise to widespread protests on the ground that the technology was obvious and had been for decades. Although the pto Commissioner himself ordered reexamination and all forty-one claims were subsequently rejected on grounds of obviousness or lack of novelty,[172] the incident highlighted the prior art problems facing the software patent industry. To the extent that the patent system thus fails to work adequately, it cannot achieve an economic balance in the stream of innovation. But these problems are not inherent in software-related patents. On the contrary, the Advisory Commission’s recommendations would go a long way toward remedying this weakness, and the pto has set out to implement many of these recommendations.[173] Hence, it seems reasonable to conclude that the patent system can be made as rational in the software industry as in other technical areas.[174]

---

are only becoming popular today due to the tremendous increase in computer power that facilitates their implementation. See statement of Lee Hollaar , pto Software Hearings (Jan. 26, 1994). In any event, the length of protection has not been a major concern in recent complaints about software-related patents. (The statutory period is being changed from seventeen years from issue to twenty years from filing as the result of recent trade negotiations.)

[172] See pto, Reexamination, BNA Patent, Trademark & Copyright J. 485 (March 31, 1994).

[173] On improvements in pto software patenting procedures, see 47 bna Patent, Trademark & Copyright Journal 357, 504 (1994). In addition, an institution known as the Software Patent Institute has been established to develop a database of software prior art, particularly unpatented prior art, to be available to the public and the pto.

[174] Much of the criticism of software-related patents goes not to the efficacy of patent examination but rather stems from the notion that the great bulk of software-related patents go to large firms and therefore the institution of software-related patents hampers small firm innovation. Whatever the incidence of patent applications by size of firm in such patent applications relative to patent applications in other industries may be, it is clear that software-related patents are sometimes crucial for small firms in maintaining their market position against larger rivals. One illustration is a recent case involving compression technology, Stac Electronics v. Microsoft Corp., No. C-93-0413-ER (C.D. Cal.), in which Stac received an award of $120million in compensatory damages against Microsoft and, in a subsequent settlement involving Stac’s claim and a successful Microsoft’s claim for misappropriation of trade secrets, Stac received financial payments, a capital infusion and a paid-up cross license from Microsoft. See Microsoft and Stac Settle, Computer Lawyer 25 (June 1994); and James W. Morando and Christian H. Nadan, Do Software Patents ‘Stac’ the Deck Against the Competition, Computer Lawyer 1 (April 1994). The Stac illustration also shows why software-related patents may be essential for small firms seeking financing of their growth; without patents, their prime assets may be their employees, who of course are not pledgable.

XI. Sui Generis Alternatives to Copyright and Patent Protection

An alternative to the copyright and patent systems for protecting software could perhaps be provided by some third system yet to be adopted. Proposals for such third systems usually are referred to under the heading of sui generis protection. The central concept behind the sui generis approach is that intellectual property protection for a particular sphere of economic activity should be by special legislation tailored to that sphere. Such proposals have been made from time to time for computer software, usually as an alternative to copyright protection. The fundamental argument for a sui generis approach is simply that since neither the copyright nor the patent systems were created with software in mind, it should be possible to create a new system that fits the special nature of software better. This argument, however, proves too much for it would lead to the creation of a separate sui generis system for every new technology, an approach rejected for nearly every new technology over the last two hundred years.[175] Copyright was created without photography in mind, and patent without biotechnology in mind. Yet it would surely make little sense to subject all new technologies to the legislative process for the creation of its own tailored system, particularly in the early years of a technology when its shape and dynamic are still unclear. Although the sui generis approach thus runs contrary to a sensible tradition, it deserves to be assessed on its own merits.

The fundamental economic problem with a sui generis approach has to do with property rights. Both copyrights and patents are property rights.[176] As with rights in tangible property, a property rights system works best when rights are specifically demarcated and circumscribed. That objective is most clearly seen with real property, where metes and bounds descriptions, plats, surveys and the like are means to achieve that objective. In the case of some intangible personal property, especially intellectual property, that objective cannot be fully achieved for a wide

[175] For exceptions see the discussion of the sui generis system for semiconducter chips, infra notes 179-190and accompanying text, and several statutes involving plants, the Plant Patent Act of 1930, 35 U.S.C. § 161et seq. and the Plant Variety Protection Act of 1977 U.S.C. § 2321et seq. The 1930 statute is perhaps not an exception for it largely applied existing patent law to certain plants and to that extent is analogous to extending copyright to new technologies. An attempt by a state to prohibit direct molding of boat hulls, arguably a kind of sui generis protection, was struck down in Bonito Boat, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141 (1989).

[176] For copyrights, see discussion supra note 74 and accompanying text. For patents, see Economic Underpinnings, 253.

variety of definitional and administrative reasons. Nonetheless, gray areas and penumbras, where the question of ownership has to be left for negotiation and litigation, reduce the certainty of ownership and therefore the economic efficiency of intellectual property rights.[177]

Some lack of definition in intellectual property rights is therefore an inevitable consequence of their very lack of tangibility. They cannot be physically measured, marked or sequestered. Thus, the real question about a sui generis approach boils down to whether a specific legislative approach or the more general judicial approach based on a more generic statute, such as the copyright and patent statutes, provides more certainty and is, in general, more efficient. Although it would probably be a mistake to prefer one over the other for all realms of human activity, there are a number of general reasons for preferring the judicial over the legislative approach, stemming essentially from differences between the adjudicatory and legislative processes.[178]

In the computer software arena this issue takes on special importance because of rapid technological change and the alleged incompetence of the judiciary to understand the underlying technology, much less the incessant changes in it. Among the factors therefore that need to be investigated in fully evaluating a sui generis approach for computer software, in addition to generic differences between the two branches, are the relative competence of Congress and the Federal judiciary to understand computer-related technologies, to adapt rules to change, and to be precise about rules in order to define and circumscribe property rights.

These large and complex issues transcend the bounds of this article, but it is worth examining the one instance where a sui generis approach to

[177] Negotiation need not necessarily be inefficient where there is a small number of parties involved, as is normally the case with real property boundaries. In software it may not be clear who the contending parties are until late in the history of a particular set of innovations. Not only may transactions costs therefore be higher but that fact interacts with the appropriability problem to increase the risk facing an innovator and hence to reduce the prospective risk-adjusted return for any prospective research and development investment. Moreover, by the time an actual dispute arises, a form of bilateral monopoly may have been created, raising transactions costs and enhancing the likelihood of litigation. While these problems all exist in principle, their importance in the software world whenever firms using competing technologies are readily identifiable should, however, not be exaggerated for the reasons discussed infra notes 106-113and accompanying text.

[178] These general reasons have been extensively discussed elsewhere. See Richard A. Posner, Economic Analysis of Law 523-536 (4th ed. 1992), and citations therein.

intellectual property protection has been used in a field of computer-related technology. That is the Semiconductor Chip Protection Act of 1984, which protects mask works used in the manufacture of semiconductor chips.[179] This statute has not been considered a great success, to say the least. Two principal reasons for its relative lack of success may be highlighted.

One reason is that the Congress tried to craft a statute that simultaneously granted protection and withheld it in a core area—reverse engineering. Not only was reproduction of mask works specifically permitted 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,"[180] but the results of that reverse engineering could be incorporated in the follow-on firm's own products.[181] So long as the follow-on firm maintained an appropriate "paper trail" concerning the steps taken in its reverse engineering to show that it had done some independent work, it could sell a chip copied heavily from the innovator's chip, thereby opening a safe passage for clever pirates.[182] It can be argued that this giving with one hand, while taking away with the other, is characteristic of detailed legislation in view of the role of conflicting interest groups, logrolling and

---

[179] 17 U.S.C. §§ 901-914. In Brooktree Corp. v. Advanced Micro Devices, Inc., 705 F. Supp. 491, 494 (S.D. Cal. 1988), aff'd, 977 F.2d 1401 (Fed. Cir. 1992), the trial court said that the Act was not sui generis legislation because it was "based upon concepts derived from the copyright law." However, the statute constituted a specific rejection of the copyright statute as the operative basis for protection and therefore for present purposes must be considered sui generis protection.

[180] 17 U.S.C. § 906(a)(1).

[181] 17 U.S.C. § 906(a)(2).

[182] Brooktree v. Advanced Micro Devices, 977 F.2d 1555 (Fed. Cir. 1992). The Brooktree case establishes a "substantially identical" standard of infringement once the appropriate paper trail is shown by the defendant. See the approved jury instructions, 977 F.2d at 1567. Rauch argues that the reverse engineering exception slowed innovation because the follow-on competitor lacks an "incentive to develop a new solution": "So long as the competitor can create a paper trail, the competitor is free to use the unique structures of the first chip. The innovator, knowing in advance that new design is likely to be appropriated, will discontinue innovative work. The return on investment can no longer be realized when the result is free for the taking." John G. Rauch, The Realities of our Time: The Semiconductor Chip Protection Act of 1984 and the Evolution of the Semiconductor Industry, 75 J. of Pat. and Trademark Office Soc. 93, 123 (1993).

other aspects of the Congressional process.[183] The fundamental question, to which no conclusive a priori answer can be given, is whether the Federal judicial process is not more likely to lead to cleaner decisions, freer of vitiating compromises. It is true that the very structure of the Federal judiciary leads to conflicts in circuits, which we now experience in copyright law between the *Whelan* and *Computer Associates* approaches to the scope of non-literal protection.[184] Even if the conflicts in circuits lead to Supreme Court review, the experience in related fields such as antitrust does not lead to confidence that the Supreme Court will set down clear rules for the future.[185] Part of the reason lies in the nature of a multi-member court,[186] but another part no doubt has to do with the relative lack of specialization in the Supreme Court's docket. The situation with respect to patent protection, where the creation of the Federal Circuit in 1982, which has become de facto the supreme court of patents, has led to some improvement in the consistency and professionalism of patent decisions.[187]

A second reason for the 1984 Act's shortcomings was that technology outpaced the Congress. Since 1984 the methods of the industry have fundamentally changed, as individual chips have become enormously more complicated and as manufacturing process technologies have consequently become less standard among firms but more crucial to success. The 1992 Patent Advisory Commission found that the "some of the basic definitions [of the Act] are already obsolete, leaving important parts of mask work technology outside the protection of that legislation."[188] Moreover, according to Rauch, whereas in 1984 simple photography of another's chip was the essential method of piracy, photography alone can no longer suffice unless the pirate has the same process technologies as the innovating firm. The result is that the pirate is able to use reverse engineering, complete with a contrived paper trail

[183] On the other hand, it should be borne in mind that even some of the innovating leaders of the semiconductor industry, such as Intel, favored the reverse engineering exception. Ibid.

[184] See discussion supra notes 70-85 and accompanying texts.

[185] See, e.g., the discussion of the continuing uncertainty with respect to the extraterritorial application of the Sherman Act in Kenneth W. Dam, Extraterritoriality in an Age of Globalization: The Hartford Fire Case, 1993 Sup. Ct. Rev. 289.

[186] There are, of course, reasons explained in the literature on public choice why nine judges may find it difficult to articulate clear rules even within majority opinions. See Frank H. Easterbrook, Ways of Criticizing the Court, 95 Harv. L. Rev. 802, 811-831 (1982).

[187] Dam, Economic Underpinnings 269-270 and authorities cited therein.

[188] Advisory Commission Report 151.

(indeed, a paper trail that is likely to be better than that of a legitimate innovator doing design work on a graphics workstation), and hence as a practical matter the 1984 Act no longer provides substantial protection.[189]

The important point is not whether Rauch is right on the facts (though his argument certainly seems plausible). Rather it is that Congress has not readdressed the issue to keep pace with technological change. Thus, the usefulness of sui generis legislation in an industry characterized by rapid technological change is called into question by the experience under the 1984 Act.[190]

A further quite practical objection to sui generis protection is that an abandonment of patent and copyright protection in favor of a new sui generis approach would leave the U.S. software industry seriously exposed abroad. National patent and copyright systems are based on a network of treaty obligations that assure more or less worldwide protection (subject to piracy and the failure of some countries to join this worldwide system). The 1992 Advisory Commission consequently concluded that the replacement of this international network, which would no longer operate to the extent that the United States abolished patent and copyright protection, by a new one based on sui generis legislation would take too long and be too uncertain to be workable: “Sui generis laws would require new, separately negotiated treaties, and international recognition of the sui generis protection would be limited or nonexistent until the lengthy process of treaty negotiation could be completed, if ever.”[191] Furthermore, the recently adopted trips agreement, negotiated in the framework of the gatt Uruguay round, requires

---

[189] Rauch, supra note 182.

[190] Posner, supra note 178 at 53 l, points out:

> This [majority vote] requirement makes legislative enactment a difficult and time-consuming process because of the transaction costs involved in getting agreement among a large number of individuals. Once a statute is passed, it is unlikely, given the press of other legislative business, to be substantially altered or repealed.

See also id. at 531 n. 2.

[191] Advisory Commission Report 151. The World Intellectual Property Organization sponsored a sui generis treaty that was agreed to by forty countries in 1989, but the United States refused to sign because of what it regarded as inadequate protection. Donald J. Chisum and Michael A. Jacobs, Understanding Intellectual Property Law § 6D[2][c] (1992). As of January 1, 1994, only ten countries had become a party and neither of the two top chip-producing countries (Japan and the United States) had become party. World Intellectual Property Organization, Copyright 14 (Jan. 1994).

signatories to maintain copyright protection.[192] Part of moving to sui generis protection would therefore have to be a renegotiation of the trips agreement.[193]

Whatever the shortcomings in the competence of general patent and copyright law to deal with complex technologies (and competence in complex technologies certainly cannot be advanced as a particular strength of the Congress), the Federal judiciary has shown not only a willingness to grapple with detailed computer software issues, but also a remarkable ability to evolve in its approach as the software industry evolves. As we have seen in one specific field, the idea/expression distinction, one can perceive three generations of judicial opinions in the relatively short period that the 1984 sui generis Chip Protection Act has languished on the statute books.[194]

XII. Conclusions

The central issue confronted in this article has been what economic analysis has to offer to the field of intellectual property protection for computer software. An economic approach to intellectual property law is no doubt in its infancy[195] and therefore an approach to protection for a particular industry, such as computer software, must necessarily be tentative. Some relatively clear conclusions nonetheless emerge. First, existing copyright and patent law provides a sound basis for an economically efficient system of protection. Second, the copyright law as currently applied deals adequately with the appropriability problem, without creating significant monopoly or rent-seeking problems. Third, copyright law provides a sound basis for preserving a balance between innovation today and innovation tomorrow, though the need for

[192] Agreement on Trade-Related Aspects of Intellectual Property Rights, 33 I.L.M. 81 (1994).

[193] A further practical objection to a sui generis approach is that now that software is a well-established industry with thousands of firms, a switch in the United States would require a costly restructuring of innumerable complex licensing and other contractual relations based on traditional copyright and patent protection.

[194] See discussion supra notes 66-74 and accompanying text. To be sure, the second and third generation precedents overlap in time but that circumstance suggests another characteristic of the Federal judicial approach—the ability through a hierarchy of courts to obtain resolution between conflicting precedents. Of course, that process also takes time, which is costly when the definition of property rights is at stake.

[195] See, however, the various articles by Friedman, Kitch, Landes, Menell, Merges, Posner, and others cited in this article and in Dam, Economic Underpinnings.

achieving such a balance does not always appear clearly in either the case law or the literature.

These sanguine conclusions concerning copyright depend crucially, however, on preservation of the distinctions between attachment and replacement and between transformative and substitutive uses, distinctions that do not always emerge sufficiently clearly in the legal literature on software.

Though patent protection for software-related inventions has substantially different coverage from software copyrights, software-related patents are found to be, in general, economically sound. However, as actually administered by the pto, the system may not adequately balance innovation today versus innovation tomorrow because it is susceptible (until important administrative changes are implemented) to generating too many invalid patents. A third alternative, sui generis protection , is briefly examined, but the experience under the 1984 statute on computer chips, as well as general considerations concerning the relative competence of the legislative and judicial branches to deal with rapidly evolving technology, raise serious questions about the desirability of the sui generis approach. A number of practical objections to enacting sui generis protection at this relatively late date are also surveyed.

This Working Paper is a preliminary version of an article that will be published in *The Journal of Legal Studies*. Readers with comments should address them to:

Kenneth W. Dam
Max Pam Professor of American and Foreign Law
The Law School
The University of Chicago
1111 E. 60th Street
Chicago, Illinois 60637

20. Alan O. Sykes, An Introduction to Regression Analysis (October 1993).
21. Richard A. Epstein, The Ubiquity of the Benefit Principle (March 1994).
22. Randal C. Picker, An Introduction to Game Theory and the Law (June 1994).
23. William M. Landes, Counterclaims: An Economic Analysis (June 1994).
24. J. Mark Ramseyer, The Market for Children: Evidence from Early Modern Japan (August 1994).
25. Robert H. Gertner and Geoffrey P. Miller, Settlement Escrows (August 1994).
26. Kenneth W. Dam, Some Economic Considerations in the Intellectural Property Protection of Software (August 1994).