Chapter 4
Controversies Over Software Protection

Legal protection for computer hardware is usually provided by patentor trade secret; this combination served fairly well to protect major hardware advances, as well as more-incremental developments. Protection for computer programs does not fit neatly within the traditional forms of intellectual property.[1] As a result, the process by which software developers and users, the courts, and policymakers have attempted to determine what should or should not be protected, and what is or is not protected, has been controversial.

LEGAL CASES

The litigation that has shaped copyright protection for software has come in three stages or ''waves. The first wave of litigation considered whether computer programs were protectable at all. This was settled by the 1980 software amendment to the 1976 Copyright Act (94 Stat. 3015, 3028), which confirmed that copyright applied to computer programs. The second wave explored which aspects of a program are protectable and which are not. Court cases have decided that program source code, object code, audiovisual (screen) displays, and microcode are protected by copyright.[3] The third and continuing wave deals with the more ambiguous aspects of what in a program is protectable (e.g., "look and feel"), and how to determine if two programs are "substantially similar."4

Copyright and patent lawsuits continue to test and explore the boundaries of the current laws. Many in industry and in the legal profession believe that if properly applied, copyrights and/or patents are adequate to protect software.[5] They argue, moreover, that sui ***generis*** approaches risk obsolescence and lack the predictability provided by legal precedent (argument by analogy to prior decisions), as well as an established treaty structure providing international protection.[7] Others consider the development of ***sui generis*** protections or significant modifications of current protection are preferable to forcing software to fit models that are suited to other types of works and discoveries but maybe ill-suited for software.[8] At the same time that some are calling for major revisions in software protection, others are arguing that the current system is not broken and

---

[1] Some observers have characterized the difficulty as due to software's being "too much of a writing to fit comfortably into the patent system and too much of a machine to fit comfortably in the copyright system." (Pamela Samuelson, "Why the Look and Feel of Software User Interfaces Should Not Be Protected By Copyright Law," *Communications of the ACM*, vol. 23, No. 5, May 1989, pp. 563- 572.)

Others consider that software's "fit" is no more uncomfortable than that of some other works, and argue that the courts can successfully apply traditional copyright principles to software cases. Anthony L Clapes, Patrick Lynch, and Mark R. Steinberg, "Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Rotation for Computer Rograms," *UCLA Law Review, vol. 34,* June-August 1987; Morton David Goldberg and John F. Burleigh, "Copyright Protection for Computer Programs," *AIPLA Quarterly Journal,* vol. 17, No. 3, 1989, pp. 2%-297.

[2] See Raymond T. Nimmer and Patricia Krauthaus, "Classification of Computer Software for Legal Protection: international Perspectives," *International Lawyer,* vol. 21, Summer 1987, pp. 733-754

[3] For a general discussion of what can be ~@@ see Cary H. Sherman, Hamish R. Sandison, and Marc D. Guren, Computer *Software Protection Law* (Washington, DC: The Bureau of National Affairs, Inc., 1989), sections 203.5(c)-203.7(c).

The copyrightability of microcode as a "computer program" was upheld in February 1989. See discussion of NEC Corp. v. *Intel* Corp. (10 USPQ 2d 1177 (N.D. Cal. 1989)) in Goldberg and Burleigh, op. cit., footnote 1, pp. 309-311.)

[4] Substantial similarity is a subjective test for copyright infringement. A plaintiff must show that the alleged infringer had access to his copyrighted *work* and that there is substantial similarity between the works at the level of protected expression. An allegedly infringing work need not be 100 percent identical to another in order to infringe its copyright, but deciding how similar works must be to prove infringement can be troublesome, even for conventional literary works like plays or novels. For computer programs, the 'ordinary observer" making the determination may need to be a technical expert. For discussion see Susan A. Dunn, "Defining the Scope of Copyright Protection for Computer Software,' *Stanford Law Review,* vol. *38,* January 1986, pp. 497-534; and Clapes et al., op. cit., footnote 1, pp. 1568-1573.

In determining substantial similarity between a copyrighted work and an accused work, courts look at the worka considered as a whole. For programs, this means the detailed design, not just individual lines of code (Clapes et al., op cit., footnote 1, p. 1570).

[5] For some discussions of these views, focusing on copyright, see Clapes et al., op. cit., footnote 1; and Goldberg and Burleigh, op. cit., footnote 1.

[6] *Sui generis* is a Latin phrase used to describe a law that is "of its own kind or class."

[7] For example, the Beme Convention provides reciprocal copyright protection in 79 countries.

[8] See, for exampk, Pamela Samuelson, "CONTU Revisited: The Case Against Computer Programs in Machine-Readable Form, *Duke Law Journal,* September 1984, p. 663-769; and Peter S. Menell, "TailoringLegal Protection for computer Software," *Stanford Law Review,* vol. 39, No. 6, July 1987, pp. 1329-1372.

does not need fixing --or at least, can be "fine-tuned" within the existing legal framework.[9]

The extent of copyright protection for the logic underlying a program, as well as its structure and interfaces, raises complex issues.[10] Some of these issues are currently the subject of well-publicized copyright lawsuits. What may be at stake in these cases is the extent to which copyright should be interpreted to give patent-like protection, especially since copyright applies for a much longer time and lacks patent's standards for novelty, nonobviousness, specificity of claim, and disclosure. Patent protection for algorithms also raises complex issues.[11] Ongoing patent suits concerning software-related inventions and the recent publicity given to some patents for algorithms have stimulated debates concerning the extent to which software-related and algorithmic inventions should be included in the patent system, and whether or not computer processes and algorithms are different enough from other technologies to warrant special provisions (e.g., shorter duration, pre-issuance notice, etc.). These debates focus on two questions:

1. the longer term question of whether patent (or patent-like) protection for software-related inventions and/or algorithms is generally desirable; and
2. the near-term questions of how well current United States Patent and Trademark OffIce (PTO) procedures are working and how to improve the comprehensiveness of the prior art available to patent examiners and private searchers (see app. A).

STAKEHOLDERS AND THEIR CONCERNS

There is a public interest in the form and level of *software protection and its effects on innovation, technology transfer, and economic growth.* At a micro level, software users have specific expectations and concerns, but they are also concerned with software quality (as for any other product) and with support and consultation for using the software. Many users consider technological anti-copying devices that curtail a program's use, or prevent modification of the software, or their ability to make backup copies as undesirable.[12] Rather than having to learn a unique set of commands and features for each program, users want to learn universal skills applicable to many programs. Cost is a factor, especially for schools or businesses that have to buy dozens of the same software package for their terminals. Devising or enforcing protections, even against literal copying by private individuals, is complicated by public sentiment that noncommercial private copying is acceptable.[13]

The software industry is concerned with unauthorized private copying and with commercial piracy. But issues that arise in one segment of the

---

[9]For example, a recent forum convened by the Computer Science and Technology Board (CSTB) began with a statement that it was not convened to challenge the legal framework for intellectual-property law, which "isn't broken and doesn't need fixing." (Lewis Branscomb, opening remarks, "Intellectual Property Issues in Software: A Strategic Forum, " CSTB, Nov. 31-Dee. 1, 1989.)

OTA NOTE: Based on discussions at the CSTB forum and comments received by OTA on a draft of this paper, the semantic dividing line between "modifications' and 'free-tuning' seems to be that the first might be interpreted to include statutory changes to copyright and patent, or sui generis forms of protection, while 'free-tuning" would imply incremental judicial refinements through specific cases. Based on reviewer comments on a draft of this paper, a substantial portion of the controversy over whether software's fit in the current system is 'neat' or 'comfortable" seems to be motivated by concerns that any discussions of less-than-perfect fit are intended to support "modification" rather than "fine-tuning."

[10]See for example: Dennis S. Karjala, "Copyright, Computer Software, and the New Protectionism," *Jurimetrics Journal, vol. 27,* fall 1987, pp. 33-%; and Peter S. Menell, "An Analysis of the Scope of Copyright Protection for Application Programs, " *Stanford Law* Review, vol. 41, 1989, pp. 1045-1104.

[11]For example, see Donald Chisum, "Patentability of Algorithms," *University of Pittsburgh Law Review,* vol. *47,* summer 1986, pp. 959-1022. Chisum argues against exclusion of "mathematical" algorithms from patentable subject matt((*Gottschalk* v. Benson) and concludes that lack of unambiguous patent protection for algorithms may ''induce attempts to rely on other sources of law, such as copyright and trade secrets, that are inherently less suited to the protection of new technological ideas with widespread potential uses" (ibid., p. 1020).

[12]Conventional wisdom is that consumer resistance led many software producers to stop copy -protecting application-software packages.

[13]A 1985 OTA survey found that the majority of respondents considered it acceptable to trade computer programs with friends in order to make copies for their own use. (U.S. Congress, Office of Technology Assessment, *Intellectual Property Rights in an Age of Electronics and Information,* OTA-CIT-*302* (Melbourne, FL: Kreiger Publishing Co., April 1986), table H-1.)

software industry may not be as important in another.[14] Therefore, policy issues must be analyzed normatively, taking different industry structures, incentives, and economics into account.

Individual Software Creators and the Software Industry

Creators of commercial software are concerned about profitability. An important rationale for intellectual-property protection for software is to give commercial software developers adequate market incentives to invest the time and resources needed to produce and disseminate innovative products. Direct revenue losses due to commercial piracy are not the only concerns of developers. Developers want to gain and maintain a competitive advantage in the marketplace. One powerful source of market advantage is lead time: the first company out with an innovative computer program benefits from its head start. Trends in software technology, like computer-aided software development, are eroding lead-time advantages. Another market advantage is user and/or machine interfaces. Here, however, the industry's goals of expanding the market and a fro's goal of maintaining market share can be at odds (see below and ch. 1, footnote 15).

There are several types of interface "compatibilities": hardware-to-user, software-to-user, hardware-to-hardware, software-to-hardware, and software-to-software (e.g., between an operating system and application programs). Compatibility and "openness" in interface standards are important to the industry as a whole. There is a concern that too much protection could raise barriers to entry for small, entrepreneurial companies if large corporations with more financial and legal resources hold key copyrights and patents. Another concern is that "bottleneck" patents or broad interpretations of copyright protection may block progress in the industry as a whole.[15]

Software competitors, and the industry as a whole, are concerned with shared access to state-of-the-art knowledge and diffusion of information about programs and programming, so that programmers can build on each others' work, rather than reinvent the wheel (or rewrite a matrix-multiplication subroutine) for each new application.[16] The pace of innovation can be speeded up if competitors are able to build on others' advances. The "PC revolution" was in large part driven by the desire to decentralize control and knowledge of computing-to bring powerful tools to the desktop of millions of users, rather than have them cloistered in the hands of a few computer specialists. Part of the "hacker ethic" and practices that produced the innovative machines and software that brought about this PC revolution were based on principles of free access and use of software and innovative techniques. Almost 15 years after the beginning of the "revolution," the "hacker ethic" is at odds with the need for income from production of software which leads developers to seek increased software protection.[17]

A major concern of most PC-software developers is private copying of an entire program by one's current or prospective customers (e.g., making an unauthorized copy of a spreadsheet program for a friend). A major concern of most vendors is literal copying of an entire program for sale by "pirate" competitors. These concerns can be dealt with fairly straightforwardly-at least in theory-by copyright law; in practice, enforcement, especially over pri-

---

[14] For example, commercial piracy is a great concern for PC-software developers; Eleventh-Amendment (States' rights) private-copying, and software-rental issues are also very important to them. The software-rental issues stem from developers' concerns that most rented software is rented to copy, rather than to "try before buying. ' PC-software developers perceive their weapons against unauthorized private copying and commercial piracy to be education, moral suasion (including "amnesties" for unauthorized users) and litigation. (Ken Wasch, Software Publishers Association, personal communication, Aug. 28, 1989.)

By contra% developers of hard-wired microcode ("firmware") are unlikely to worry about private individuals making copies at home, at least with presently available technology.

[15] The need for at least some degree of compatibility for 'network' technologies like softw are-whether through informal (de facto) industry standards of formalized ones--i s an important consideration in making policy choices about desirable levels of protection and how these are achieved. Some consider that extending copyright protection to user interfaces and the "look and feel" of programs might lead competitors to offer incompatible but otherwise similar products ("locking in" users to particular product lines), rather than competing on price and performance features of an industry-standard product. On the other hand, these types of protection could lead to competition in product design, producing major advances. (See Joseph Farrell, "Standardization and Intellectual Property," *Jurimetrics Journal*, vol. 30, No. 1, fall 1989, pp. 35-50.)

[16] For example, it may be wasteful duplication of effort to have to create an entirely new user interface each time a program is written.

[17] See Steven Levy, *Hackers: Heroes of the Computer Revolution (Garden City, NY:* Anchor Press/Doubleday, 1984), especially ch. 2.

vate copying or overseas piracy, is difficult.[18] Copying software is easy and inexpensive. Moreover, private copying of software seems as "natural" as making home audiotapes or videotapes to many individuals, and allows them to avoid expensive purchases. Some individuals and businesses engage in commercial piracy, making and selling unauthorized copies of software.[19]

The legal status of some software-engineering practices is not clear under copyright. Some practitioners think that 'clean room' reverse-engineering procedures might be acceptable practices under copyright because a second program that is developed "independently" without access to the protected expression in a prior program does not infringe the copyright of the previous one.[20] This is controversial, however, because clean-room practices vary.[21] Some reverse-engineering steps like de-compiling object code (or dis-assembling assembly-language code) in order to analyze the program's functions generally involve making one or more copies of the code as an intermediate step in the process of creating a "new" one.[22]

Software Users

Millions of individuals and thousands of businesses rely on purchased software products for their day-to-day activities and livelihood. They care about the price, quality, functionality, ease of use, and variety of software products available. Thus, they care about the health of and level of competition in the software industry. They also want "common ground' (compatibility) that allows them to use new products with their existing hardware and software. Users care about having "reasonable" rights (e.g. being able to make a backup copy of an expensive piece of software); some need the ability to modify "packaged' software in order to use it efficiently or meet other specialized needs.

Most businesses and individuals who use software tools to create other products or services want a stable and predictable legal environment so they know what uses are permitted and which are not, and which must be licensed from developers. A 1989 survey of nearly 200 management-information system (MIS) executives showed that almost one third reported that 'look-and-feel" lawsuits will cause them to shy away from software clones.[23] legal uncertainties about patented computer processes may have a similar effect, particularly because patents ' "use" rights affect the buyer as well as the developer.

The "software work force" who use and/or create software as part of their jobs want to have transferable skills; thus they are concerned, sometimes only indirectly, with standards for programming languages and external consistency of user interfaces. (For example, learning a new word-processing package is easier if it has commands and functions similar to other packages one already knows.) But users also want more powerful software with improved functions. Sometimes consistent ("standard") interfaces can conflict with ease of use and improved functionality.[x]

[18] A rule of thumb in the software industry is that at least one unauthorized copy exists for every authorized sale of a computer program. Some software publishers think the number of unauthorized copies is even higher—from 3 to 7 for every legitimate copy sold. (Estimate by the Software Publishers Association cited in Peter H. Lewis, "Cracking Down on Software pirates," *New* York *Times,* July 9, 1989, p. F1O.)

[19] Estimates of losses vary and reports of losses may be somewhat eve- because it is not clear that each unauthorized COpy displaces a Sale. The Software Publishers Association (SPA) estimated that PC-software producers lost about $1 billion in sales to "piracy" (defined to include both copying for personal use and copying for commercial profit) in 1986. The Lotus Development Corp. estimates that over half ($160 million) of the potential sales for its Lotus 1-2-3 package are lost every year. Micropro International estimated that it lost $177 million in potential sales for Wordstar in 1984, compared to $67 million in actual revenues. (Industry estimates cited in Anne W. Branscomb, "Who Owns Creativity? Property Rights in the Information Age," *Technology Review,* vol. 91, No. 4, May/June 1988.)

[20] See Arizona State University College of Law, Center for the Study of Law, Science, and Technology (Milton R. Wessel, Director), "The 'Structure, Sequence and Organization' and 'Look and Feel' Questions," LaST Frontier Conference Report, June 1989, pp. 8-12.

[21] In one version, a software-development team reads the source code of a program and writes a description of its functions (i.e., extracts the ideas from the expression). The source code may have been obtained by reverse-compiling or reverse-assembling object code. The first team's functional description is passed to a second team, which designs a new program without "contamination" from the original code.

[22] The recent decision in *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 109 S. Ct. 971,9 U. S.P.Q. 2d (BNA) 1847 (1989) has raised controversy over the Supreme Court's likely view of reverse engineering of computer programs (seethe articles of D. C. Toedt, Arthur Levine, and Allen R. Grogan in *The Computer Lawyer,* vol. 6, No.7, July 1989, pp. 14-36). Sherman et al., op. cit., footnote 3, Sec. 210.8, question the validity of a "clean room" defense to a claim of infringement.

[23] Michael Alexander, "Criticism Builds Over Impact of Look-and-Feel Litigation," *Computerworld,* vol. 23, No. 18, May 1, 1989, p. 14.

[24] See Jonathan Grudin, "The Case Against User Interface Consistency," *Communications of t& ACM,* vol. 32, No.10, October 1989, pp. 1164-1173.

Academic Community

Academic research communities value free access to and exchange of information. Academic software and computer-science researchers and developers, motivated by other than commercial potential (e.g., professional prestige, tenure, publication in scholarly journals), tend to view intellectual-property protection somewhat differently than do commercial developers. However, universities and their faculties are increasingly interested in commercializing technology and obtaining revenue for use of their intellectual property.

Many in the academic community are concerned that what they see as "over-protection" (such as copyright protection for "look and feel' and patenting of software processes and algorithms) might hamper research and long-term growth in their fields. Some believe that the artistic expression of a user interface should be protected, but not the way commands are invoked at the user interface. They believe that forcing developers to contrive meaningless variations in interfaces solely to avoid legal entanglements will hinder software research and development. [25]

Software is used by students and educators in all disciplines. [2b] Cost, quality, and variety are important, and educational institutions face difficult problems in providing equitable student access to software (e.g., 22,000 university students may each need access to 500 dollars' worth of software-how should this be accomplished?) .27 This and other issues like ethical software use in education, are the focus of a joint project by the EDUCOM software initiative (EDUCOM is a nonprofit consortium of 650 colleges and universities) and ADAPSO (the computer software and services industry association). [%] In contrast to major commercial software packages, faculties in a number of disciplines develop "small" software programs to help teach students. The incentives to develop and use 'small' software differ significantly from those for commercial software, as do the means of distribution (e.g., over academic computer networks). [29]

SOME PRIVATE EFFORTS TO SORT THINGS OUT

In March 1989, several members of the legal and software-development communities met at an MIT Communications Forum session on software patenting. [30] The session focused on the PC-software industry. Participants reviewed the history of software development and patentability, and stated different views about the merits of software patents and their effects on innovation and creativity.

In February 1989, the Arizona State University College of Law, Center for the Study of Law, Science, and Technology convened a group of conferees to identify areas of agreement in the legal academic community concerning copyright principles for computer software. [31] The conferees reached consensus on several points:

- Courts will have to adapt traditional copyright principles to a new and different technology .32
- The phrase "structure, sequence, and organization" is unhelpful to describe expressive elements of programs. It does not distinguish

[25] Alexander, op. cit., footnote 23. Grudin (op. cit., footnote 24) offers opposing views.

[26] Some students and educators may use 'educational software' programs, which are like books in that they convey information, albeit interactively. They may use "professional" or "business" software programs for graphics, numerical calculation (number-crunching), and word processing. They may also use 'discipline-specific" software (often created with Federal funding) for research and problem-solving in fields like physics, mathematics, biology, engineering, economics, geography, and architect.

[27] Dana Cartwright, Syracuse University, personal communication, Aug. 30, 1989.

[28] See for example, "Using Software: A Guide to the Ethical and Legal Use of Software for Members of the Academic Community,' EDUCOM and ADAPSO, 1987; and "can 'Intellectual Property' Be Protected?" Change (special issue), May/June 1989.

[29] Steven Gilbert, EDUCOM, personal communication, Dec. 11, 1989.

[30] Massachusetts Institute of Technology Communications Forum, "Software Patents: A Horrible Mistake?' (Cambridge, MA: Seminar notes, Mar. 23, 1989). The panel consisted of: Daniel Bricklin (Software Garden, Inc.), Stephen D. Kahn (Weil, Gotshal & Manges), Lindsey Kiang (Digital Equipment Corp.), Robert Merges (Boston University School of Law), Pamela Samuelson (University of Pittsburgh School of Law), R. Duff Thompson (WordPerfect Corp.), Brian Kahin (moderator), and Gail Kosloff (rapporteur).

[31] LaST Frontier Conference Report, op. cit., footnote 20, The conferees were Donald S, Chisum (University of Washington), Rochelle Cooper Dreyfuss (NYU), Paul Goldstein (Stanford), Robert A. Gorman (University of Pennsylvania), Dennis S. Karjala (Arizona State University), Edmund W. Kitch (University of Virginia), Peter S. Menell (Georgetown University), Leo J. Raskind (University of Minnesota), Jerome H. Reichman (Vanderbilt University), and Pamela Samuelson (Emory University/University of Pittsburgh). Others from the academic and business communities attended parts of the conference as presenters or observers.

[32] Ibid., p. 2.

expressions from processes or procedures. Moreover, computer programs are functional works, thus technological constraints on using them limits the scope of available protection.[33]

- Courts have extended copyright protection beyond the exact text of a work.[34]
- Achieving compatibility between programs that serve as software-to-software or hardware-to-software interfaces is a legitimate goal for software competitors.[35]
- Some programdevelopment practices that extract logic and use it in developing another program do not infringe copyright.[36]
- Copyright law provides a mechanism for protecting user interfaces, but the protection should be limited so that, for example, aspects that optimize in a way that has no "viable substitute" (i.e., are functionally optimal) are not protected.[37]

In other important areas, consensus was not reached:[3g]

- The extent to which copyright law protects interface aspects that are not "functionally optimal" (see last item above).
- The extent to which human factors analysis can be relied on to determine the scope of copyright protection.
- What the optimal level of software protection is.
- If a *sui generis* protection regime is desirable.

ssIbid, p. 6.
341bid

351bid, p, 7,

[36]Ibid., pp. 8-11. Conferees believed that limited copying for purposes of examinati"on and study of a program's unprotected elements (including disassembly or decompiling to get pseudo-source code from object code) would fall within the terms of fair use.

[37]Ibid., pp. 12-17.

[38]Ibid., pp. 2-17.