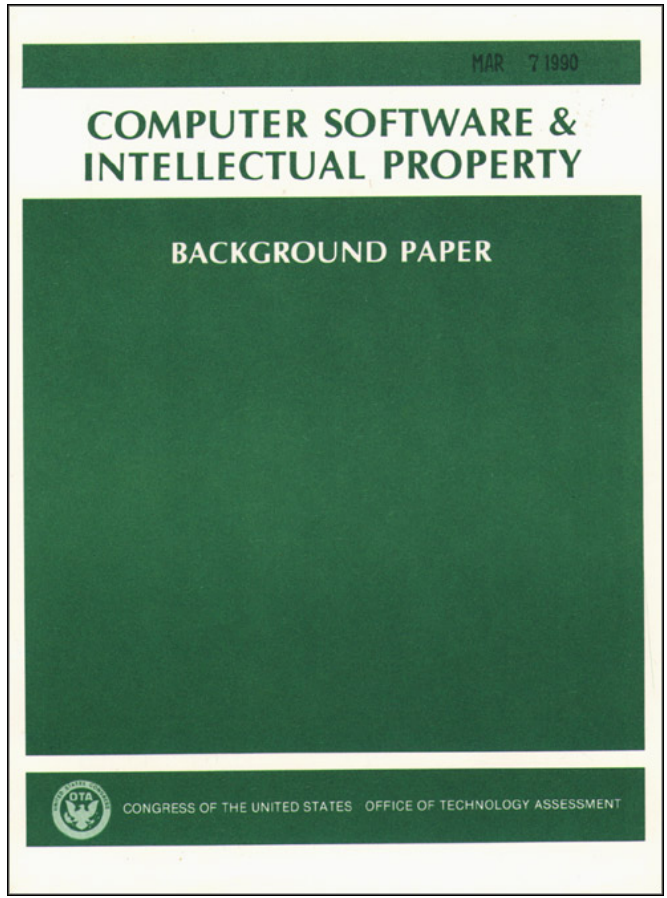
Computer Software and Intellectual Property

March 1990

OTA-BP-CIT-61
NTIS order #PB92-169242





Recommended Citation:

U.S. Congress, Office of Technology Assessment, ***Computer Software and intellectual Property--Background Paper, OTA-BP-CIT-61*** (Washington, DC: U.S. Government Printing Office, March 1990).

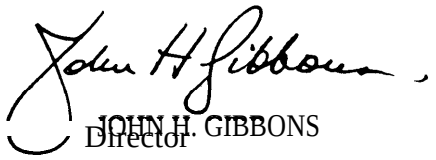
For sale by the Superintendent of Documents
U.S. Government Printing Office, Washington, DC 20402-9325
(order form can be found in the back of this report)

Foreword

In the past, programming was viewed as a support activity for computer hardware or as a hobby for "hackers." As the software industry matured, it has become less driven by technology and more concerned with the needs of users and the demands of the market. Today software is a lucrative industry of its own, amounting to some $60 billion per year in domestic sales and services. Internationally, the United States dominates the software market, holding the edge in innovation over Western Europe, Japan, and the Soviet Union. Accompanying this growth and maturity has come concern about the amount and type of intellectual-property protection available for software.

This background paper examines existing intellectual-property protection for computer software-copyrights, patents, and trade secrets—and provides an overview of the often conflicting views and concerns of various stakeholders. It was prepared in response to a request from the Subcommittee on Courts, Intellectual Property, and the Administration of Justice of the House Committee on the Judiciary.

OTA gratefully acknowledges the contributions of the many experts, within and outside the government, who reviewed or contributed to this document. As with all OTA publications, however, the content is the responsibility of OTA and does not necessarily constitute the consensus or endorsement of reviewers or the Technology Assessment Board.



JOHN H. GIBBONS
Director

Reviewers and Other Contributors

Reviewers

Anne W. Branscomb
Program on Information Resources Policy
Harvard University

Dam E. Cartwright, III
Academic Computing Services
Syracuse University

Joseph Farrell
Department of Economics
University of California at Berkeley

Francis D. Fisher
Cambridge, MA

Steven W. Gilbert
EDUCOM

Gerald Goldberg
U.S. Patent and Trademark Office

Brian Kahin
Science, Technology and Public Policy program
Harvard University

Michael S. Keplinger
U.S. Patent and Trademark Office

Ronald S. Laurie
Irell & Manella

Peter Menell
Georgetown University Law Center

Steven J. Metalitz
Information Industry Association

Eric Schwartz
U.S. Copyright Office

Lee Skillington
U.S. Patent and Trademark Office

Oliver R. Smoot
Computer and Business Equipment
Manufacturers Association (CBEMA)

Douglas R. Weimer
Congressional Research Service
American Law Division

Milton R. Wessel
Georgetown University Law Center

Other Contributors

Kenneth B. Allen
Information Industry Association

Ronald Palenski
Association of Data Processing
Service Organizations (ADAPSO)

Ron Reiling
Digital Equipment Corp.

Pamela Samuelson
University of Pittsburgh and
Emory University

Kenneth A. Wasch
Software Publishers Association

Ingrid A. Voorhees
Computer and Business Equipment
Manufacturers Association (CBEMA)

OTA Reviewers and Contributors

Karen G. Bandy
Communication and Information
Technologies program

D. Linda Garcia
Communication and Information
Technologies Program

Elizabeth Miller
Communication and Information
Technologies program

Kevin O'Connor
Biological Applications Program

Fred W. Weingarten
Communication and Information
Technologies Program

Robert Weissler
Industry, Technology, and
Employment Program

Fred B. Wood
Communication and Information
Technologies Program

NOTE: OTA appreciates and is grateful for the valuable assistance and thoughtful critiques provided by outside reviewers and other contributors. These individuals do not however, necessarily approve, disapprove, or endorse this background paper. The paper is the sole responsibility of OTA, not of those who so ably assisted us.

OTA Project Staff-Computer Software and Intellectual Property

John Andelin, ***Assistant Director, OTA***
Science, Information, and Natural Resources Division

James W. Curlin, ***Manager***
Commuication and Information Technologies Program

Project *Staff*

Joan D. Winston, Policy ***Analyst***

AM M. Hironaka ***Research Assistant***

Administrative Staff

Elizabeth Emanuel, ***Administrative Assistant***

Jo Anne Price, ***Secretary***

Karolyn St. Clair, ***Secretary***

Contents

Page

Chapter 1: Summary, Overview, and Issues 1
Overview 1
Questions for Consideration 3
Questions About Definitions 3
Questions About Industry Structure and the Nature of Innovation 3
Questions About Protection and Enforcement 3
Chapter 2: Changing Technical and Market Environments 5
Chapter 3: The Intellectual Property Bargain and Software 7
Copyright 7
Patent 8
Trade Secret 9
Chapter 4: Controversies Over Software Protection 11
Legal Cases 11
Stakeholders and Their Concerns 12
Individual Software Creators and the Software Industry 13
Software Users 14
Academic Community 14
Some Private Efforts To Sort Things Out 15
Chapter 5: International Issues 17
Appendix A: Legal protection for Computer Software 19
Appendix B: International Protection for Computer Software 25

Chapter 1
Summary, Overview, and Issues

The health and vitality of the software[1] industry are crucial to the computer industry, to government, and to the economy as a whole. In 1988, domestic revenues for software and related services amounted to about $60 billion. Over the past 30 years, software costs have increased as a share of total information-system costs. Software development costs today amount to over half of the cost for new systems.[2] Software is a critical component in the successful operation of the computer system; after all, without software, computers would be unusable. Software is vital to defense and civilian agency operations, and to industrial sectors as diverse as telecommunications, electronics, transportation, manufacturing, and finance.[3] The United States has 70 percent of the world software market, but this may be in jeopardy in the future as other countries' software industries develop.

Legal protections for computer software can affect the pace of technological advance in software and the extent to which these advances are disseminated and used in the economy, as well as affect developments in the computer-hardware industry. Software protections affect the "openness" of standards and interfaces, which are important components of firms' competitive strategies in both the software and hardware industries. Thus, the economic implications of under-protecting or over-protecting software extend far beyond the software industry alone.

This study draws on prior and ongoing OTA assessments: *Intellectual Property rights in an Age of Electronics and Information* (April 1986), *Copyright and Home Copying: Technology Challenges the Law* (October *1989), Information Technology R&D: Critical Trends and Issues* (February *1985),* and *Information Technology and Research* (ongoing).

This background paper reviews copyright, patent, and trade secret protections; discusses current issues regarding legal protection for computer software; and identifies some of the normative and positive questions that Congress should consider in its continuing oversight of computers, software, and intellectual property.

OVERVIEW

Basic questions about the detailed implementation of intellectual-property protection for soft*ware-what to protect? how much? for how long? against what? fiom whom ?—are* difficult to answer. Software does not fit comfortably into the traditional intellectual-property frameworks of copyright (which protects works of authorship)[4] or patent (which protects inventions).[5] This problem is shown in the current round of "look and feel" copyright suits and in the controversy over patent protection

[1] This paper uses the term "software" to refer to sets of instructions-computer programs—for computers, whether these are stored in punched cards, magnetic tape, disks, read-only memory (ROM), random-access memory (RAM), semiconductor chips. or on Paper.

Sometimes the *'software" is taken to mean data sets, documentation, and training support as well as programs (see "Software Technology," Nov. 6, 1989, attachment to U.S. Congress, Office of Technology Assessment, "Intellectual-Property Protection for Computer Software," Staff Paper, Nov. 2, 1989). In some ways software and databases are merging, and in the future it may be hard to distinguish between a program and its data. (For example, the "data" for some artificial-intelligence programs are themselves logical rules and structures.) However, as used here, "software" does not include electronic databases (see ch. 2, footnote 12 for more on databases).

[2] When software maintenance costs are added, software costs can amount to 90 percent of total costs over the life of an information system. (Barry W. Boehm,&@wureEng ineering ***Economics (Englewood*** Cliffs, NJ: Prentice-Hall, Inc., 1981), p. 18, cited in "Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation," staff study by the Subcommittee on Investigations and Oversight, Transmitted to the Committee on Science, Space, and Technology, U.S. House of Representatives, Aug. 3, 1989.)

[3] For example, software is critical to telecommunications. In 1965, the software in a telephone switching machine consisted of about 100,000 lines of code. Today, switch software can have over 2 million lines of code. This pattern of increasing size and complexity is similar for PBX hardware and modems. (Eric E. Sumner, "Telecommunications Technology in the 1990s," ***Telecommunications, vol.*** 23, No. 1, January 1989, pp. 37-38.)

[4] A 1980 amendment to the Copyright Act of 1976 made explicit provisions for computer programs as (literary) works of authorship (Public Law %-517, 94 Stat. 3-15, 3028). This followed recommendations made by the National Commission on New Technological Uses of Copyrighted Works (CONTU).

[5] The statutory subject matter of a patent is limited to a process, machine, *article* of manufacture, or composition of matter that is novel, nonobvious, and useful, or to new and useful improvements to these classes of patentable subject matter.

The Supreme Court hasnot ruled whether computer programs per se are patentable subject matter, but has ruled that computer-implemented algorithms that are deemed "mathematical" algorithms per se are not statutory subject matter. Federal courts have thus held that a computer processor algorithm is statutory subject matter unless it falls within a judicially determined exception like the one for "mathematical algorithms" per se. (See U.S. Patent and Trademark Office, "Patentable Subject Matter: Mathematical Algorithms and Computer Rograms," 1106 O.G. 4, Sept. 5, 1989).

for inventions involving computer programs and algorithms.[6]

Software is not unique in this respect. New technologies have challenged traditional intellectual-property frameworks before.[7] Often, traditional protection devices have been able to accommodate new technologies successfully. For example, the first copyright statute dealt only with maps, charts, and books, but copyright has been able to deal with the "hard questions" posed by works like engravings, musical compositions, photographs, and so forth.[8] But some commentators believe that new electronic technologies (including software) pose more severe challenges to copyright, in part because it is increasingly difficult to extract and freely use ideas that are communicated only in the form of expressions conveying intellectual-property rights.[9]

Another problem in determining where software fits in the intellectual-property system is that computer software and hardware technologies are changing rapidly, both qualitatively and quantitatively. This makes the crafting and refining of software protections akin to aiming at a target that isn't there yet (or doesn't yet exist). Each time one controversy or set of questions is resolved, another arises.[10] For example, future advances in computers and computation, especially in artificial intelligence and interactive computing, will require a change in the definitions of "software" and "data": a new type of computer, called a "neural net," is not programmed as are conventional computers; instead, it is "trained." It is becoming harder to distinguish between a program and the data on which it operates: expert systems are designed to draw on a knowledge base of detailed information about an area of application (e.g., medical diagnostics, industrial processes) in order to make "decisions." The knowledge base or "data" for these artificial-intelligence programs are themselves logical rules and structures, not just numerical values.

A third problem is compounding the problem of rapid technological change. The legal and technical communities do not have consistent definitions for terms like "algorithm" or "interface" that make up computer and computational parlance. For example, one common technical definition of the term algorithm is: "a set of rules which specify a sequence of actions to be taken to solve a problem."[11] But other definitions are also used within the technical community: some computer scientists consider algorithms to be simply abstract computer programs, and believe that distinctions between algorithms and programs only capture differences in degrees of abstraction.[12] Without agreement on a common language and definitions, protection issues become extremely difficult.

Finally, the international scope of software markets complicates matters, requiring domestic laws to be harmonized with treaty obligations and laws of other nations.[13] Although the United States is still

[6] In this paper, OTA *sometimes* uses phrases like "patents for software-related inventions," "software-related patents," or "patenting algorithms" to refer generally to patent protection for computer-implemented processes and algorithms. The United States Patent and Trademark Office (PTO) considers terms like "software patents" to be a misnomer because they maybe interpreted to mean that computer programs per se (i.e., the sequence of coded instructions itself) are patentable, as opposed to the underlying computer processesthey carry out---see previous footnote. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18, 1989, pp. 1-2.)

[7] These have included Phonorecords (sound recordings), motion pictures, reprography, audio and videocassette recorders, and genetic engineering. For a discussion of the latter challenge--the issue of patenting living organisms U.S. Congress, Office of Technology Asessment, New Developments *in Biotechnology: Patenting Life-Special Report, OTA-BA--370* (Washington, DC: U.S. Government Printing Office, April 1989).

[8] See Anthony L. Clapes, Patrick Lynch, and Mark R. Steinberg, "Silicon Epics and Binary Bards: Determining *the* Proper Scope of Copyright Protection for Computer Programs," *UCLA Law Review, vol. 34,* June-August 1987, pp. 1493-1594, esp. pp. 1495-1499.

[9] See Francis Dummer Fisher, "The Electronic Lumberyard and Builders' Rights: Technology, Copyrights, Patents, and Academe," *Change,* vol. 21, No. 3, May/June 1989, pp. 13-21.

Some believe that looking at software and other types of intellectual property in isolation will not prove satisfactory. Instead, they suggest that the changing nature of information expressions, and their communication and use must be examined broadly, along with economic incentives for creating and disseminating intellectual property. (Francis D. Fisher, personal communication, Dec. 8, 1989; see also Anne W. Branscomb, "who Owns Creativity? Property Rights in the Information Age," *Technology Review,* vol. 91, *No. 4, May/June* 1988, pp. 3945.)

[10] Some current software-copyright controversies involve making the distinction between (protected) expression and (unprotected) idea Future techno-legal controversies might involve works "authored" by advanced artificial-intelligence systems. (Milton Wessel, Georgetown University Law Center, personal communication, Nov. 28. 1989.)

[11] *Chambers Science and Technology Dictionary,* Peter M.B. Walker (cd.) (New York, NY: W & R Chambers, Ltd., 1988), p. 23.

[12] For a computer scientist's perspective on legal confusions resulting from unsuitable models for algorithms and computer programs, see Allen Newell, "The Models Are Broken, The Models Are Broken!" *University of Pittsburgh Law Review, vol. 47, No. 4, summer 1986, pp. 1023-1035.*

[13] Multilateral copyright *treaties* like Berne can provide simultaneous protection for computer software in many countries. Relatively few countries provide patent protection for sofhvare-related inventions. In any even~ patents usually provide potcction only in the country where issued

the leader in software development, European and Japanese competitors are advancing rapidly, especially in targeted areas like artificial intelligence. The prospect of unified markets and standards in Western Europe after 1992 poses a significant competitive challenge for U.S. software developers. One example of how intellectual-property protections for software will help shape competition in these new international markets is in their influence on standards[14] and interfaces. If the way a program interfaces with people (user interface), interfaces with other programs (software interface), or interfaces with computers (machine interface) is protected, it will be more difficult for industry to agree on standard conventions to make programs compatible with one another.[15] Standards and interfaces will help determine the extent to which various countries', as well as different companies', software and hardware are compatible. The degree of compatibility will shape the future of global information networks, and will determine the ease of access.

QUESTIONS FOR CONSIDERATION

In its oversight of policies to protect computer software and related technologies, Congress may find the following questions helpful.

Questions About Definitions

- Terms like "interface" and "algorithm" (or "mathematical algorithm") do not have uniform meanings for the computer and legal professions. What terminology can be developed or adopted to discuss and analyze software issues, so that the legal and software professions, and policymakers, can meet on common ground?
- How can it be ensured that the definitions used and distinctions sought will be meaningful as technology changes?
- In what ways are functional works like computer software and algorithms different from other types of works and inventions? In what ways are they similar? Can software be examined apart from other types of electronic information?
- For software and other forms of electronic information, is it useful to talk about policies to "reward and compensate" producers, rather than to "protect" their intellectual property?

Questions About Industry Structure and the Nature of Innovation

- Does it make sense to refer to "software" ox the "software industry" in aggregate? What are the different types of software and segments of the industry? Should some be treated differently?
- Where and how has innovation in software occurred? Who creates new software techniques? Commercializes or disseminates them? Is this changing?
- Does the current statutory scheme of copyright and patent protection adequately stimulate creativity and innovation in software? If so, can it be assumed that "what worked before will work in the future"?
- Does the current scheme create sufficient economic incentives for investment in software research and development? For commercialization of R&D results? If so, will it continue to do so?
- Is the current scheme sufficient to maintain U.S. leadership in software in a world market?

Questions About Protection and Enforcement

- What aspects of software and/or algorithms should be protected?
- Do concepts like "lead time" have a different meaning for software or algorithms than for other

[14] Standards mechanisms differ in the United States and abroad. In the European Economic Community, standard development almost always concerns de jure standards. In the United States, the term "standard" is most often used to mean "de facto (voluntary) standard" or "dominant product." (Oliver Smoot, Computer and Business Equipment Manufacturers Association (CBEMA), personal communication, Dec. 7, 1989.)

[15] "Standardization" of interfaces in different application software packages is an issue because users find common interfaces attractive when these allow their current hardware and software to be compatible with new products or make learning how to use new software easier.

Some software developers want their programs to have user interfaces (e.g., the way commands are invoked, or "look and feel"), software interfaces (e.g., degree of data portability between programs), or machine interfaces (e.g., operating systems needed to run the programs) similar to or in common with others' programs in order to gain a larger potential market. But other developers, such as those who are first to market a radically innovative program, may see their interfaces as critical parts of their competitive advantage, These developers may want to protect their interfaces in order to reap economic rewards for developing them.

types of works and inventions? What does this imply for the duration needed for protection?

- How feasible will enforcement of protections for software and/or algorithms be? Will courts be able to draw the distinctions needed?
- Where will the burden of proof be in enforcing rights? Will they fall equitably on individuals, large firms, and small firms?
- Does "fair use" need to be interpreted differently for software than for other types of copyrighted works? Are special rules needed for uses of software (as opposed to other types of works and technologies) in education and research?
- Who speaks for the public interest in issues involving computer software and other forms of electronic information?

Changing Technical and Market Environments

Changes in computer hardware and software technologies and markets have shaped concerns about protection for computer software and ideas about what kinds of protection are needed.

Computer hardware technologies have changed dramatically over the past decade. With these changes have come important changes in how software is developed, sold, and used. Consequently, some software developers have modified their ideas about what aspects of software need the most protection. For example, as writing and checking lines of program instructions ("code") becomes more automated through computer-aided software development, some software producers propose to protect the *logic and idea* of a program, not just the effort required to write code and check ("debug") it. Others are concerned that computer-aided software development will make it easier to "disguise" copying.

Technological change also challenges traditional copyright concepts. For example, with developments in artificial intelligence and in interactive software and database systems, it will likely become increasingly difficult to draw the line between derivative works[1] and new creations, and to determine what constitutes "authorship."[2]

Twenty-five years ago, computers and software were not mass-marketed, retail items. The mainframes and minicomputers of the day were relatively few in number, compared to the number of microcomputers (PCs) in use today. These machines were run by expert staff using expensive, often custom-developed, software.[3] In the late 1960s, the "independent" software industry began to flourish. By 1988, U.S. independent software developers' revenues exceeded $25 billion, up from $20 billion in 1987.[4] About 40 percent of these revenues were from foreign sales.[5] Domestic revenues from all software and related services totaled over $50 billion in 1987, and were expected to increase to about $60 billion for 1988.[6] The United States currently commands a 70 percent share of the world software market.[7]

The fortunes of computer-software and computer-hardware developers are closely intertwined. A computer may gain popularity if plentiful and/or novel software is available. Conversely, lack of suitable application software (programs designed to perform specialized tasks for users) can be a barrier to the market success of a new computer or can limit the effective use of a computer.[8] Scarcity of application software can impede the use of a whole class of computers: software to make most effective use of massively parallel processors and other supercom-

---

[1] A "derivative Work" is a work based on one or more preexisting works (e.g., a translation, abridgement, or other form of transformation or adaptation). (See Title 17, U.S.C. 101.) Section 117 allows the rightful owner of a piece of software to make a copy or adaptation if the new copy or adaptation is for archival purposes (a "backup" copy) or is an essential step in utilizing the program in the computer.

[2] For some interactive software, it is increasingly difficult to determine where the programmer's expression ends and the user's contribution begins-the computer mediates and intermingles the creative efforts of both. Interactive computer-based works may generate new questions about ownership and originality. See discussion and example of a hypothetical interactive music-composition program, "Minstrel," in 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), pp. 70-73.

[3] Although some relatively sophisticated users (e.g., in universities or research organizations) did develop and maintain their own programs, most application software for specific tasks like inventory control or number crunching was either provided by hardware manufacturers or custom-developed under contract. Almost all operating-system software to run the computer and control its input, output, and logic functions was provided by computer-hardware manufacturers.

[4] Association of Data Processing Service Organizations (ADAPSO) figures on industry performance, 1989. These data for "non-captive" firms excludes the value of software produced in-house by hardware manufacturers; revenues are split about evenly between application and operating-system software.

[5] U.S. International Trade Commission, "The Effects of Greater Economic Integration Within the European Community on the United States," July 1989, ch. 4, p. 39.

[6] Computer and Business Equipment Manufacturers Association, *The Computer, Business Equipment, Software and Services, and Telecommunications industry, 1960-1996* (Washington, DC: CBEMA, Industry Marketing Statistics, 1987), table 4-3, p. 99.

[7] Commission of the European Communities, "Green Paper on Copyright and the Challenge of Technology-Copyright Issues Requiring Immediate Action," June 1988, pp. 171-172.

[8] Because software has become so critical t. so many industrial sectors, while productivity growth for software technology has been relatively slow, there is some concern that software could become a bottleneck-or the "Achilles heel of the information age." (Ian M. Ross, President, AT&T Bell Laboratories, keynote address, 1988 Bicentennial Engineering Conference, Sydney, Australia, Feb. 23, 1988.)

puters is currently scarce.[9] On another front, application-software developers can find the existing "installed base" of older computers and earlier programs (e.g., spreadsheet, database, or word-processing programs) a barrier to adoption of new programs designed for more advanced machines. They may also need to upgrade their products periodically; these new versions must be compatible with new hardware and also with older versions of the product.[10]

System software (programs, including operating systems, that make the computer usable and control its performance) can be an important factor in hardware firms' competitive strategies. For example, product competition in PC markets is based in part on differences in system *features (e.g.,* processing speed, ways of shipping data for processing in different parts of the computer, graphics capabilities) and *user-interface features (e.g.,* pictoral "icon," manual "mouse," or keystroke "macro" commands for functions such as moving the cursor or saving a file). These advantages are acquired from shrewdly mixing hardware and software designs.

When Congress created the National Commission on New Technological Uses of Copyrighted Works (CONTU) in 1974, the "PC revolution" had not yet begun to bring desktop computing power to the millions of individuals that now use it. By the time CONTU issued its final report in 1978, the PC revolution was under way, creating anew generation of computer users who were not primarily programmers or computer experts. The rapid proliferation of PCs in homes, offices, and schools created a very large retail market for application software-for word processing, spreadsheets, even games-as well as a lucrative market for PC operating-system software. In 1988, domestic revenues for PC application software reached almost $3 billion.[11] The widespread use of PCs also facilitated the growing use of online databases.[12]

Rapid growth and technological innovation made markets for PCs and PC software quite volatile, compared to the mainframe and minicomputer markets a decade earlier. Some new hardware and software firms would introduce ,new products, enjoy brief success, then go out of business within the space of a few years. Other firms built on early successes and went on to become industry leaders. A few years after introducing a successful product, however, they might find a substantial fraction of their potential market taken by competitors offering similar-sometimes improved—products, often at a lower price. The volatility of PC markets has focused new attention on questions about how best to provide intellectual-property protection for software, as well as hardware. At the same time, the history of the computer hardware and software industries illustrate the complex relationship between intellectual-property protection and stimulation of creativity.

---

[9] A recent press briefing by the Institute of Electrical and Electronic Engineers reported that while U.S. supercomputer manufacturers are focusing on new hardware developments to stay ahead of Japanese competitors, they are giving little attention to software to exploit the hardware's speed and power. As a result, a supercomputer's speed in solving problems may be only 1 to 2 percent of its advertised peak speed. ("Software SolutionJ'Science, vol. 246, No. 4930, Nov. 3, 1989, pp. 574-575.) See also: "The Computer Spectrum," *Computer*, vol. 22, No. 11, Nov. 11, 1989, pp. 61-62.

[10] Successive generations of upgrades tend to be increasingly complex. For example, one software developer's first database-management package had several thousand lines of code and took a single developer less than a year to create. The most recent version, designed to accept data files created under earlier versions of the package, has hundreds of thousands of lines of code and has taken a team of developers several years to create. (Ruthann Quindlen, "installed Base Becoming Obstacle toSoftware Companies' Success,'' *Infoworld*, vol. 11, No. 36, Sept. 4,1989, p. 82.)

[11] Ann Stephens, Software Publishers Association, personal communication, Oct. 2, 1989.

[12] An "electronic database" is a collection of information stored and accessed by electronic means. (Commission of the European Communities, "Green PaperonCopyright and the Challenge of Technology-Copyright Issues RequiringImmediateA ction," June 1988, p. 205.) Databases can be copyrighted as *works of compilation; the* copyright extends to the material contributed by the author of the compilation, or to the author's creative efforts in selecting, ordering, and arranging preexisting material, not to the preexisting material per se or ideas included in the compilation. Domestic revenues for on-line business databases alone amounted to $6.5 billion in 1988 (Information Industry Association data, 1989).

The Intellectual Property Bargain and Software

In the United States, an *"intellectual property bargain"* underlies the concept of intellectual-property protection. This "bargain" between creators and society balances two social objectives: 1) it encourages the production and dissemination of new works and inventions (by providing economic incentives to creators), and 2) it promotes access to and use of these works and inventions.[1] Thus, the limited monopoly granted to authors by copyright and to inventors by patents is a quid-pro-quo arrangement to serve the public interest, rather than a system established primarily to guarantee income to creators. (See app. A for reviews of copyright, patent, and trade secret protections as they pertain to software.)

COPYRIGHT

Copyright is granted to authors for the creation of certain classes of works.[2] The economic underpinnings of copyright assume that to profit from a work, the author will publish or otherwise disseminate it to the public.[3] Copyrights, which are relatively easy to obtain and long-lasting compared to patents, are intended only to protect the ***expression*** in a work from unauthorized copying, not to protect the underlying ideas or functionality from use. Even "expression" is not protected from independent creation.

The recommendation by the National Commission on New Technological Uses of Copyrighted Works (CONTU) that copyright protection be explicitly extended to all forms of computer programs was established in the 1980 amendments to the Copyright Act.[4] Even then, CONTU recognized certain difficulties in applying copyright (which does not protect ideas, processes, or procedures) to software, which is inherently functional. A particular concern was the impossibility of establishing a precise line between the copyrightable "expression" in a program and the noncopyrightable processes it implements—the distinction between "expression" and "idea."[5] CONTU assumed that most copyright infringements in the then-immediate future would be "simply copying," but recognized that technological advances would raise more difficult questions in determining the scope of copyright.[6] CONTU concluded, however, that these questions should be answered on a case-by-case basis by the Federal courts.[7] Many continue to believe that traditional copyright principles should continue to be applied to software, because difficulties in distinguishing between idea and expression are not unique to software and because copyright law has been able to embrace many new forms of authorship within existing principles.[8]

The 1986 OTA report, *Intellectual Property Rights in an Age of Electronics and Information,* discussed the increasing difficulties of applying copyright to *functional works* such as programs.[9] Some of these difficulties are shown today by ongoing "look and feel"[10] and "structure, sequence,

[1] See U.S. Constitution, art. I, sec. 8, cl. 8. For a discussion of this bargain and the public interest in intellectual property protection, see U.S. Congress, Office of Technology Assessment, Intellectual *Property Rights in an Age of Electronics and Information,* OTA-CIT-302; and *Copyright and Home Copying: Technology Challenges the Law,* OTA-CIT-422 (Washington, DC: U.S. Government Printing Office, October 1989), ch. 3.

[2] Title 17, U.S.C. 102(a).

[3] See OTA-CIT-302, op. cit., foomote 1, p. 7 and ch. 6. Copyright inheres in a work as soon as it is created and also exists for unpublished works.

[4] CONTU recommended that programs be protected as literary works. CONTU's definition of ● 'computer program" was added to Sec. 101 of the Copyright Act of 1976 and a new Sec. 117 was added limiting computer-program copyright holders' exclusive rights.

[5] See Peter s. Menell, "An Analysis of the Scope of Copyright Protection for Application Programs," *Stanford Law Review,* vol. 41, 1989, p. 1047.

[6] "Fi~ Report of the National Commission 011 New Technological Uses of Copyrighted Works," July 31.1978. pp. 22-23.

[7] Ibid., p. 23.

[8] For discussion of this view, see Morton David Goldberg and John F. Burleigh, "Copyright Protection for Computer Programs," *AIPLA Quarterly Journal, vol.* 17, No. 3, 1989, pp. 294-322. Goldberg and Burleigh argue that the courts have (as Congress intended) conscientiously applied traditional copyright principles to software cases and, for the most part, are reaching prep= and well-feasoned results (ibid., p. 296).

[9] See OTA-CIT-302, op. cit., footnote 1, pp. 78-85. The 1986 report identified three types of copyrightable works: works of art, created for their own *intrinsic value; works* of fact, such as databases, whose value lies in an accurate representation of reality; and works of function, such as computer programs, which use information to describe or implement a process, procedure, or algorithm.

[10] "Look" is often taken to mean the appearance of screen displays, "feel"to mean the way the program responds when the user selects options or enters commands. User interfaces, including graphic icons or combinations of keystrokes to represent functions like "save" or "delete," are part of "look and feel."

and organization" copyright suits. Moreover, market changes, like the almost-hundredfold increase in PC use since CONTU, make the financial stakes much higher.

PATENT

Copyright protects the expression of an idea. Patent protects the ***technological application*** of an idea in a machine or process.[11] A patent precludes "practice" of the invention (e.g., making, using, or selling the claimed invention) by others, even if they invent it independently. But the requirement for patentability is stringent: the invention must be useful, novel, and nonobvious compared to prior discoveries (the "prior art") that are patented, in the public domain, or otherwise widely known.[12] While publication is not required for copyright, patent is granted in exchange for full *disclosure* of what the inventor considers the best way of implementing or practicing the invention.[13] The purpose of the patent is to "teach" others and thereby stimulate technological progress as they seek to build on (or invent around) the discovery.

The availability of patent protection for software-related inventions was unclear (generally considered "not applicable") until the early 1980s.[14] Since 1981, there has been renewed interest in patents for software-related inventions.[15] Over the past 7 years, patents have been issued for software-related inventions such as linear-programming algorithms, spell-checking routines, logic-ordering operations for spreadsheet programs, brokerage cash-management systems, and bank college-savings systems.[16]

In the last year, some patent lawsuits concerning software-related inventions and controversies concerning patents for algorithms have become highly visible. These lawsuits and specific controversies have focused *concerns* over the appropriateness of patent protection for software-related inventions and algorithms. These concerns arise both from lack of belief that patents in computer-program processes encourage technological progress, as well as from the practical problems that software-related inventions and algorithms raise for patent-system administration.

One of these problems is the incomplete stock of "prior art" available to patent examiners in evaluating patent applications for processes involving computers, especially those involving software and

[11] The statutory subject matter of a patent is limited to a process, machine, article of manufacture, or composition of matter that is novel, nonobvious, and useful, or to new and useful improvements to these classes of patentable subject matter.

For an overview of patents, including a discussion of criteria for patentability and how a patent is obtained, see U.S. Congress, Office of Technology Assessment, *New* Developments *in Biotechnology: Patenting Life—Special Report,* OTA-BA-370 (Washington, DC: U.S. Government Printing Office, April 1989), ch.3.

[12] Although all "original" programs are generally eligible for copyright, the fraction of programs potentially able to qualify for patent protection is much smaller. For one thing, the U.S. Patent and Trademark Office (pTO) position is that computer programs per se are not patentable, as opposed to patentable computer processes and algorithms (see footnote 15 below).

In the early 1980s, some commentators estimated that over 90 percent of computer-program inventions would not in principle meet the patent requirement that the invention be nonobvious, compared to the prior art. Therefore, they estimated that patent protection would only be relevant to about 1 percent of all software. (Findings of ABA Proprietary Rights in Software *Committee* (1983), cited in Cary H. Sherman, Hamish R. Sandison, and Marc D. Guren, *Computer Software Protection Law* (Washington, DC: The Bureau of National Affairs, Inc., 1989), pp. 401-408 and note 41.) (OTA NOTE: The 90 percent and 1 percent figures do not refer to the percentage of patent applications that result in a patent being issued.)

[13] The specification disclosed illustrates one implementation of the invention; others may be possible. The patent application must describe the invention adequately to allow a person of "ordinary" skill (in the particular area of technology) to make and use the invention.

[14] In 1972, the Supreme Court stated *that* certain inventions performed by computers could be patentable subject matter (*Gottschalk v. Benson,* 409 U.S. 63 (1972)), A 1981 Supreme Court decision (*Diamond v. Diehr, 450* U.S. 175 (1981)) helped clear the way for patent protection for some software-related inventions by clarifying the circumstances under which inventions performed by computers could be patentable subject matter.

[15] The Supreme Court has not ruled as to whether computer programs per se constitute patentable subject matter. Currently, PTO patent examiners carry out a two-part test for mathematical-algorithm statutory subject matter; the test is intended to be consistent with legislative history and case-law. For examination purposes, "mathematical algorithms" are considered to refer to "methods of calculation, mathematical formulas, and mathematical procedures generally," and no distinction is made between man-made mathematical algorithms and mathematical algorithms representing discoveries of scientific principles and laws of nature (which have never been statutory subject matter). For a process claim involving a mathematical algorithm to be patentable, the claim excluding the algorithm is required to be statutory subject matter—i.e., the claim must be for a process, machine, etc. Trivial post-solution activity like displaying a number is not sufficient. ("Patentable Subject Matter: Mathematical Algorithms and Computer Programs," 1106 O.G. 4, Sept. 5, 1989; also contained *in Patent Protection for Computer Software: The New Safeguard,* Michael S, Keplinger and Ronald S. Laurie (eds.) (Englewood Cliffs, NJ: Prentice Hall Law and Business, 1989), pp. 9-42.)

[16] In this paper, OTA sometimes uses phrases like "patents for software-related inventions, ""soflvvare-related patents," or "patenting algorithms" to refer generally to patent protection for computer-implemented processes and algorithms. The U.S. Patent and Trademark Office (PTO) considers terms like "softwarepatents" to be a misnomer because they may be interpreted to mean that a computer programmer se (i.e., the sequence of coded instructions itself) is patentable, as opposed to the underlying computer process it carries out. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18, 1989, pp. 1-2.)

algorithms.[17] The published literature does not completely embody the development of the fields of software and computer science. In many cases, important prior art exists only in product form and is not described in print form such as articles in technical journals.[18] Another problem is the lack of special classifications or cross-references to issued patents. As a result, it is virtually impossible to find, let alone count or profile, all software-related or algorithmic patents. This means that patent examiners and the public have no effective way of searching and studying such patents.

Another problem is the long time lag between patent application and issuance, compared to quick-moving software life cycles. Someone may develop and bring a software package to market, unaware that it will infringe on a patent applied for by another developer, but not yet granted. These are called "landmine patents," and can occur in other areas of technology besides software.[19]

TRADE SECRET

Trade secret protection, provided under individual State laws, protects against use or willful disclosure of the secret by others (but not against independent discovery). Most foreign nations outside of Western Europe do not have extensive trade-secret laws. However, most developed countries do have some form of legal protection for confidential business information[20] and contracts or licenses can often provide equivalent protection abroad.

Trade-secret information maintains its status so long as the information is not publicly disclosed.[21] Unlike copyright or patent, there is no limitation on its duration. Trade secret has been the favorite mechanism to protect mainframe and minicomputer software and its underlying ideas, logic, and structure because programs are licensed to specific customers, not the mass market. Mass-marketed PC software is sometimes released with "shrink wrap" licenses intended to maintain trade-secret status (see app. A). Software that is protected effectively as a trade secret does not become prior art. This can adversely affect patent examinations and lead to "reinventing the wheel."

---

[17] Interest in patenting software-related inventions and algorithms is relatively new. Copyrighted software deposited at the Copyright Office is not readily searchable for patent purposes. Also, trade secrecy has been a major form of software protection, and trade-secret information may not constitute part of the prior art, Therefore, the prior art readily searchable by patent examiners and the public has gaps. This potentially allows patents to issue for computer-process inventions that are already known in the industry or that represent only minor improvements. However, the PTO is working to improve the file of prior art for search purposes.

[18] Ronald Laurie, Irell & Manella, personal communication, Dec. 21, 1989.

[19] However, some observers believe that timing poses special problems for software-related inventions because of a combination of factors 1) the decentralized nature of the software industry, 2) difficulties in determining the prior art, and 3) the rapid rate of software-product development and short product life cycles, compared to the time required for processing a patent application. (Brian Kahin, Harvard University, personal communication, Dec. 1, 1989; and Brian Kahin, "The Case Against 'Software Patents'," personal communication on Dec. 1, 1989.)

[20] Michael Keplinger, Gerald Goldberg, and Lee Skillington, Patent and Trademark Office, personal communication, Dec. 18, 1989.

[21] To maintain trade-secret protection for software, developers may require that employees or transferees hold the information in confidence.

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.

Chapter 5
International Issues

Software is an important positive part of America's position in international trade. A study by the United States International Trade Commission (ITC) estimates that in 1987 almost 40 percent of U.S. software developers' revenues came from foreign sales.[1] Indirectly, computer software contributes to the efficiency of other businesses and manufacturers competing in international commerce.[2]

The global nature of the software industry must be recognized when considering domestic intellectual property protection. For example, U.S. treaty obligations under the Berne Convention, Universal Copyright Convention, and Paris Convention mean that domestic laws will protect foreign fins, along with domestic firms, in the U.S. markets If U.S. law differs substantially from international norms of copyright and patent protection, U.S. software producers may find it difficult to have their claims for intellectual property protection recognized in foreign countries.

Intellectual property law is important to encourage and to protect U.S. works and inventions internationally. The United States is attempting to include intellectual property in the General Agreement on Tariffs and Trade (GATT) treaty and is engaged in bilateral negotiations as well. (App. B reviews mechanisms for international intellectual-property protection and looks at some issues concerning international competition and trade.)

As the software industry evolves on an international scale, intellectual-property issues will continue to grow in importance. Currently, the United States is in the forefront of software development. However, we must be sensitive to shifts in the world economy, such as the changes in the European Economic Community proposed for 1992. As global networks develop, hardware and software standards will also become more important.

Piracy abroad can reduce the economic incentives to invest in software development and can give rise to diplomatic and trade problems.[4] Lack of adequate intellectual-property protection abroad makes it more difficult to protect U.S. works and inventions in foreign markets, while strong software protection in the United States benefits both foreign and domestic producers. Lack of protection might also complicate North-South technology transfer to less-developed countries (LDCS) and East-West transfer to Eastern Europe and the People's Republic of China. In some of these countries, commercial software piracy has become ingrained, making software companies less willing to make state-of-the-art software available.[5] Many of the nations where commercial piracy is widespread are Third World countries, who may be trying to develop a computer industry of their own or who cannot afford to pay full price for software. U.S. producers, however, lose revenues through this piracy, and may be unable to develop legitimate markets in these countries.

---

[1]U.S. International Trade Commission, "The Effects of Greater Economic Integration Within the European Community on the United States," July 1989, ch. 4, p. 39.

[2]As one commentator notes, "Information technologies are fast becoming the raw material of the global economy. . .[n]ew information technologies are changing the way the manufacturing sector conducts business just as radically as they are changing the character of the service industries. The manufacturing sector is relying more and more on services as inputs, including R&D, engineering, sales, accounting, finance, and even management." (Clarence J. Brown, 'The Globalization of Information Technologies," *The Washington Quarterly,* winter 1988, pp. 90, 95.)

[3]Thus, strong U.S. laws benefit foreign competitors; as foreign software suppliers grow stronger, this may become more important. For further discussions of international conventions and a lengthier treatment of other international issues, see app. B.

[4]Worldwide, piracy for computer software and hardware is estimated to have cost producers $4 billion in 1986 (this figure is based primarily on firms' own estimates of losses). (Estimate based on a study performed by the U.S. International Trade Commission, *Foreign Protection of Intellectual Property Rights and the Effect on U.S. Industry and Trade,* February 1988, table 4-1, p. 4-3.)
The International Intellectual Property Alliance estimated that software piracy in 11 "problem" countries amounted to $547 million in 1988. (International Intellectual Property Alliance, "Trade Losses Due to Piracy and Other Market Access Barriers Affecting the U.S. Copyright Industries: A Report to the United States Trade Representative on 12 'Problem Countries'," April 1989, p. viii.)

[5]For example, the People's Republic of China has no copyright law of its own (although it is currently drafting one with provisions for software) and is not a member of international conventions. One study sponsored by several industry groups has estimated that software piracy in the People's Republic of China cost U.S. developers some $300 million in 1988. (international Intellectual Property Alliance, op. cit., footnote 4.)

Appendix A
Legal Protection for Computer Software

Computer **software** can be protected under copyright patent or trade secret law, or under some combination of these. This appendix briefly reviews these forms of protection, with emphasis on applications to computer software.

A related, sui generis, form of protection for semiconductorchip mask designs is provided via the Semiconductor Chip Protection Act of 1984.[1] The Act protects the designs of the mask works used to lay out integrated circuit designs in semiconductor chips.[2]

Copyright

The current copyright law is enacted in the Copyright Act of 1976, as amended (Title 17 U. S.C., ch. 1-8,90 Stat. 2541). A 1980 amendment made explicit provisions for computer programs as (literary) works of authorship (Public Law %-517, 94 Stat. 3-15, 3028). Copyright protects "original works of authorship" from ***unauthorized uses*** including reproduction (copying), making derivative works (adaptation), public distribution, public performance, and display.[3] Generally, the term of copyright for new works is the life of the author plus 50 years, or 75 years for works made for hire (e.g., by art employee of a firrn).[4]

Copyright has been the form of software protection favored by most nations (see app. B). Obtaining a copyright is easy, inexpensive, and quick compared to the requirements for obtaining a patent (see next section on patents). Since copyright is administered under Federal law, unlike trade secret protection, it is uniform in all the States. The duration of copyright protection is very long, compared to the expected economic or technical lifetimes of computer programs.

The doctrine of fair use is one of several statutory limitations on copyright holders' exclusive rights. Under this doctrine, certain unauthorized uses, such as copying for the purposes of teaching, scholarship, or research, may be considered "fair use:' not copyright infringements. Whether an instance of copying is a fair use instead of an infringement is determined by the courts, taking four statutory criteria into account: 1) the purpose and character of the use, 2) the nature of the copyrighted work, 3) the amount and substantiality of the portion used in relation to the work as a whole, and 4) the effect of the use on the potential value of or market for the work.[5]

Another statutory limitation on the rights of software copyright holders is given by section 117 of the copyright law, added in the 1980 amendment:

> . . . it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:
>
> 1. that such new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other reamer, or
> 2. that such new copy or adaptation is for archival Purposes Only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.

This limitation clarifies the right of a user who legitimately owns a software product to make "backup" copies of the software to protect against damage or loss, to load the software onto the hard disk of a computer for easier or more efficient use, and to make adaptations if necessary to make the program usable on a computer (e.g., compiling it, inserting default formats or directory paths,

[1]Public Law 98-62(1,98 Stat. 3347,3356.

[2]See Cary H. Sherman, Hamish R. Sandison, and Marc D. Guren, *Computer Software Protection Law* (Washington, DC: The Bureau of National Affairs, Inc., 1989), part 300. Protection for an original mask work extends for 10 years from the time it is registered; the duration takes into account the relatively short useful economic life of a particular chip. The Act was developed in a period when chip design and mask-work production was a very labor-intensive and time-consuming step in chip manufacture, so that protection from copying of the mask work conferred a significant competitive advantage. The Act does not provide protection for the underlying idea, or against independent creation, reverse engineering, or instances where the design was not copied. Patent protection also applies to chips, but patents require public disclosure of the invention. Also, the layout of a chip will rarely satisfy the levels of novelty and nonobviousness required by patent law, which protects invention, not effort.

[3]An "original work" is one that does not have the same expression as a preexisting work; an identical, but independently created, work is not a copyright infringement. (The "originality necessary to support a copyright merely calls for independent creation, not novelty." Melvin B. Nimmer, N*immer on* Copyright (New York, NY: Matthew Bender, 1982), vol. 1, sec. 201(A), cited in U.S. Congress,Office of Technology Assessment, *Intellectual Property Rights in an Age of Electronics and Information* (Melbourne, FL: Kreiger Publishing Co., April 1986), OTA-CIT-302, ch. 3, footnote 10.) chapter 3 of the 1986 report discusses intellectual property concepts.

A "derivative work" is a work based on one or more preexisting works, such as a translation, abridgement, condensation, etc. (Copyright Act, sec. 101).

[4]See Copyright Act, sec. 302. For further discussion of U.S. copyright law, see U.S. Congress, Office of Technology Assessment, *Copyright and Home Copying: Technology Challenges the Law*, OTA-CIT-422 (Washington, DC: U.S. Government Printing Office, October 1989), ch. 3.

[5]Copyright Act of 1976, ch. 1, sec. 107. Computer software can present some particular problems in assessing what is "fair use." For example, a competitor who de-compiles a copyrighted object-code program in order to study the unprotected ideas it embodies will necessarily make a "copy" of the entire program (see app. A, footnote 7).

etc.). It does not permit making and distributing multiple copies for school or office use.

Copyright does not confer rights over ideas-only the *expression* of an idea is protected, not the underlying idea itself.[6] A copyright holder (e.g., a software developer) might consider this to be a disadvantage, because his copyright will not preclude a competitor from creating a new work embodying the same idea, so long as the competitor does not incorporate copyrighted expression from the first program into the second program. For software, copyright may also allow "clean room" reverse-engineering practices.[7] In this type of reverse engineering, one team of software developers studies the code of a copyrighted program to extract the underlying functionality (ideas). A second team (who has never had access to the copyrighted code) then creates a new program, based on the first team's functional specifications. For some computer software, writing the code may be relatively trivial, so that the true innovation and market advantage lie in the program's logical structure or in its interfaces. The extent to which these are protectable expression, as opposed to uncopyrightable ideas, is the focus of the latest round of court cases.

Disputes Over Copyrightability

There has been considerable disagreement over what features of a computer program are (or should be) copyrightable. The distinction between idea and expression can be very tricky to make, even for some traditional literary works like books and plays. For software, which is intrinsically functional, idea and expression are closely interwoven, even in theory. In practice, it is extremely difficult to separate which elements of a program are the expression and which are the underlying idea.[8] There is substantial disagreement among legal scholars and among software developers and computer scientists as to whether copyright should protect only against literal or near-literal copying (e.g., mechanical translations, paraphrasing, and disguised copying), or should also protect a program's structure, sequence, and organization and user interfaces (including "look and feel") as well.[9] For example, some in the computer and legal professions believe that a program's "look and feel" should not be protected by copyright instead, these individuals think that protection for "look and feel" is better suited to a patent framework.[10] Others, however, are critical of patent protection for computer processes in programs.[11]

Many in the computer and legal professions believe that "traditional rules of copyright law adapt very comfortably" to new forms of expression like computer latest programs.[12] These individuals believe that there is considerable room for expression in even detailed aspects of a program like its design, logic, structure, and flow,[13] and that the courts can be and generally have been successful in adapting traditional copyright principles to software-infringement cases, even those involving idea-v.-expression or structure, sequence, and organization questions.[14] Therefore, they think that copyright is viable—and vital-as a vehicle for protecting computer

[6] "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." (Copyright Act of 1976, ch. 1, sec. la(b).)

[7] One court decision has found copying for the purpose of reverse engineering to be sanctioned by section 117 of the Copyright Act. (*Vaults* v. *Quade*, 655 F.Supp. 750, E.D. LA. (1987), affirmed, 847 F.2d 255 (1988), cited by Brian Kahin, personal communication, *Dec.* 1, 1989).

[8] For example, the decision in *Whelan Assoc. Inc.* v. *Jaslow Dental Laboratories, Inc.* (797 F.2nd 1222 (3rd Cir. 1986), cert. denied, 107 S. Ct. 877, 1987) held that the underlying purpose of a program is its "idea," and everything else is expression, given that more than one way to achieve the purpose is possible. Under this interpretation, virtually any elements of the program's structure, sequence, or organization would be considered copyrightable. By contrast, the decision in *Plains* Cotton *Coop*. Assoc. v. *Goodpasture Computer Service, Inc.* (807 F.2nd 1256 (5th Cir.), cert. denied, 108 S, Ct. 80, 1987) held that only line-by-line program design or literal code were protectable. (David C. Godbey, "Comment: Legal Documents As A Metaphor for Computer Programs in Copyright Analysis—A Critique of *Whelan and Plains Cotton*," *The Computer Lawyer*, vol. 6, No. 8, August 1989, pp. 1-10.)

[9] Recent court decisions have varied in determining the extent to which program structure, sequence, and organization should be protected by copyright.

The term "look and feel" originated in an article that focused attention on software user interfaces (Jack Russo and Douglas K. Derwin, "Copyright in the 'Look and Feel' of Computer Software," *The Computer Lawyer*, vol. 2, No. 2, February 1985). There is no statutory or case-law definition, although a kindred phrase, "total concept and feel, ''has been adopted by appellate courts, (Pamela Samuelson, "Why the Look and Feel of Software User Interfaces Should Not Be Protected By Copyright Law," *Communications of the ACM*, vol. 32, No. 5, May 1989, pp. 563-572.)

[10] The argument is that "look and feel" is more idea and concept than expression. In that case, however, an innovation that did not represent a novel and nonobvious advance over prior work would not be patentable (see section on patents). Thus, most user-interface improvements would not be protected under either patent or copyright if "look and feel" is not accepted by tlcourts. See: "Computer Scientists Protest SoftwareLitigation," *International Computer Law Adviser*, June 1989, p. 22; and Pamela SamuelSon, ibid.

[11] For example, see Brian Kahin, "The Impact of Software Patents," *Educom Review*, winter 1989, pp. 28-31.

[12] For a discussion of this position and a rebuttal of opposing views, see Anthony L. Clapes, Patrick Lynch and Mark R, Steinberg, "Silicon Epics and Binary Bards," *UCLA Law Review*, vol. 34, June-August 1987, pp. 1493-1594 (seep. 1501).

[13] See Clapes et al., ibid., pp. 1549-1558.

[14] See Clapes et al., ibid., especially pp. 1546-1554 and 1575-1584. See also Morton David Goldberg and John F. Burleigh, "Copyright Protection for Computer Programs: Is the Sky Falling?" *AIPLA Quarterly Journal*, vol. 17, No. 3, 1989. pp. 2%-297. Goldberg and Burleigh argue that even if not all court cases have been correct or clearly articulated, the same is true ofpatent cases forsoftware-related inventions and would be true for any *sui generis* forms of protection (ibid., p. 2%).

software, and that arguments for hybrid or sui generis protections are based on faulty premises.[15]

Before the current copyright law (and 1980 amendment), there was considerable disagreement as to whether programs could be copyrighted as writings and, if so, what forms of computer software were copyrightable-e.g., whether only the higher-level-language (or "source") code could be copyrighted, as opposed to code in assembly language or machine language (the "object" code). Some arguments-which may have distracted attention from more fundamental issues—were based on the presumed inability of humans to read lower level languages or binary object code; according to this rationale, only higher level languages expressed "writings" (for human readers) eligible for copyright protection.[16] These arguments were misguided because human programmers can and do read programs, albeit with more difficulty, in assembly language and machine language.[17]

The 1980 amendment with reference to programs as statements used "directly or indirectly" in a computer (sec. 101) and "adaptations" for purpose of use in a computer (sec. 117), as well as the explicit 1976 provisions for works that can be perceived/reproduced/communicated "either directly or with the aid of a machine or device" (sec. 101), resolved much of this confusion. Court cases have held that computer programs source code, object code, microcode,[18] flow charts, and audiovisual screen displays-are protected.[19]

Patent

A patent protects an invention, ***including application of the underlying idea,*** from copying and from independent creation for a period of 17 years. It protects against literal infringement (making, using, or selling the claimed invention) and also against infringement by equivalent inventions, whether or not the infringing inventor had prior knowledge of the patented invention. The statutory subject matter of a patent is limited to a process, machine, article of manufacture, or composition of matter that is novel, nonobvious, and useful, or to new and useful improvements to these classes of patentable subject matter.

The requirements for a ***patentable*** invention are relatively stringent; patents don't reward hard work per se. The patent requirements for novelty and nonobviousness are a finer screen than the "originality" criterion of copyright. (All "original" software is eligible for copyright, as with any other statutory work of authorship, and copyright inheres in a work as soon as it is created.) Although patents are being granted for software-related inventions,[20] only a small fraction of software is likely to contain a computer process meeting the tests of novelty and nonobviousness.[21]

---

[15] See Clapes et al., op. cit., footnote 12, especially pp. 1501-1505, 1548-1561, and 1583-84. See also Goldberg and Burleigh, ibid., pp. 317-322.

[16] The rule that a work must be readable by a human audience had its *origins* in *White-Smith Music Publishing Co.* v. *Apollo Music Co.*, 209 US. 1 (1908) which ruled that player piano rolls could not be copyrighted. For a discussion of the readability requirement see "Copyright Protection of Computer Program Object Code, ''Harvard Law *Review,* vol. 96, May 1983, pp. 1723-1744., Christopher M. Mislow, "Computer Microcode: Testing the Limits of Software Copyrightability," Boston *University Law Review, vol.* 65, July 1985, pp. 733-805., and the dissent of Commissioner Hersey in the National Commission on New Technological Uses of Copyrighted Works (CONTU), Final Report, July 31, 1978.

[17] A source program is the program as written by *the* programmer. Writing in lower-level languages like assembly language can be tedious, so programmers usually use a higher-level language like Fortran. For example, a Fortran instruction to add an input "V" to a variable "SPEED" would be ***SPEED = SPEED+ V.*** A Fortran program must be compiled before it is executed by the computer; the compiler translates each Fortran instruction into many binary machine-language instructions.

Similarly, a program written in assembly language must be assembled before it is executed. An assembly-language program generally consists of symbolic statements, each one of which corresponds to one basic operation of the computer. For example, to add "V" to "SPEED" would require statements like LD ***R0,SPEED*** (load SPEED into Register O), LD ***R1,V*** (load V into Register 1), ***AD R0,R1*** (add contents of Register 1 to Register O). Assembly-language programs can't be directly understood by the computer, so an assembler has to translate them into machine language.

In the 1950s-1960s, computer programs were usually entered in the computer in the form of punched cards. As this "source" deck was keypunched, the 80 characters of code on each card were printed at the top for verification and debugging purposes. When the program was compiled, the resulting "object" deck contained only punched holes. This may have contributed to the assumption that object-code programs could not be read by humans.

[18] Microcode governs the operation of the computer within one cycle of the computer's internal clock; itis part of the computers operating system. Copyrightability of microcode was upheld in *NEC Corp.* v. *Intel Corp.* (645 F. Supp. 590 (N.D. Cal. 1986) vacated, 835 F.2d 1546 (9th Cir. 1988).

[19] Sherman et al., op. cit., footnote 2, sees. 203.5(c)-203.7(c).

[20] In the United States, certain types of computer-implemented processes and algorithms can be patented. The Supreme Court has not ruled on whether computer programs per se are patentable subject matter, but has ruled that computer-implemented algorithms that are deemed "mathematical algorithms" per se are not statutory subject matter. Federal courts have thus held that a computer processor algorithm is statutory subject matter unless it falls within a judicially determined exception like the one for''mathematicat algorithms" per se. (See U.S. Patent and Trademark Office, "Patentable Subject Matter: Mathematical Algorithms and Computer Programs," 1106 O.G. 4, Sept. 5, 1989).

In this paper, OTA sometimes uses phrases like "patents for software-related inventions,"" software-related patertts," or "patenting algorithms" to refer generally to patent protection for computer-implemented processes and algorithms. The U.S. Patent and Trademark Office (pTO) considers terms like "software patents" to be a misnomer because they may be interpreted to mean that a computer program per se (i.e., the sequence of coded instructions itself) is patentable, as opposed to the underlying computer process it carries out. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18, 1989, pp. 1-2.)

[21] One estimate from the World Intellectual Property Organization places the fractional 1 percent. (Cited in Ingrid M. Arckens, "Obtaining International Copyright Protection for Software: National Laws and International Copyright Conventions," *Federal Communi cations Law Journal,* vol. 38, August 1986,, p..285.)

An advantage of patent protection for the discoverer of a software-related invention is that the patent will protect all the claims for the invention, taken as a whole. (Many of these processes would likely not be protectable under copyright because they would be considered part of the unprotected "idea.") A single computer program may consist of a number of patentable processes and algorithms. At the same time, the claimed invention might be executed by a number of copyrighted programs. Depending on how carefully claims are constructed, the computational logic and processes-even the algorithm itself—Carl be protected.

The United States Patent and Trademark office (PTO) issues patents on inventions that are determined to meet statutory requirements (see above), the first of which is statutory subject matter. Because of the judicially created exception for "mathematical" algorithms (see footnote 20), computer processes that are solely "mathematical" algorithms- "mathematical" ***algorithms per se—are*** not considered to be statutory subject matter. However, ***software-related inventions claiming a new or improved process*** (which can include an algorithm, perhaps even a "mathematical" one) can be statutory subject matter if the patent claims excluding the "mathematical" algorithm are otherwise statutory. Therefore, they can be patented if the other requirements of novelty and nonobviousness are m e t[z]

From the viewpoint of the software industry and society as a whole, some unattractive elements of patents for software-related inventions are procedural, and have to do with the "prior art" and patent searches. The prior art is the body of publicly known technical information against which the patentability of an invention is evaluated. Even if a discovery is "novel" compared to the prior art it must also be "nonobvious." This means that if the "differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art,' then the invention is not patentable.[23]

In principle, prior art consists of inventions previously known, sold, or used, including those described in other patents and published articles (see 35 U.S.C. 102). In practice, prior art is most often previously issued patents. Because the bulk of software continues to be protected by copyright and/or trade secret because much of the history of software development is not in the published literature and because relatively few patents for software-related inventions were granted prior to the 1980~ the available and locatable prior art is less complete and relatively more difficult to compile or search than the prior art for other technical fields. PTO classifies patents for software-related inventions according to the field of the process chimed, making it difficult to find or track patents for computer-program processes. Also, nonpatent prior art for software-related inventions is often in nonwritten form, existing only as software products. Thus, there is more risk that invalid patents may issue for widely known or "obvious" computer-program processes.

Patents under examination are not disclosed, so a competitor may put considerable effort into developing a program that unknowingly duplicates computer processes for which one or more patents are pending. The problem of timing and product life cycles is not unique to the computer and software industries, but it is especially troublesome in industries as fast-paced as these. Finally, the process of getting a patent is expensive and lengthy, compared to copyright or trade secret protection. Although turnaround time in PTO is decreasing, a patent still may take years to issue in an industry where products have short economic lifetimes. Enforcement can be difficult and time-consuming, and litigation for infringement runs the risk of finding one's patent invalid.

Problems With Terms and Models

Use of the term "algorithm"[25] has been subject to controversy, largely because computer scientists, lawyers, and the courts have used different definitions of computer-related terms and different models of how "programming" is done. Often, the legal definitions or

[22]The Supreme Court has not ruled as to whether computer programs per se constitute patentable subject matter. Currently, PTO patent examiners carry out a two-part test for mathematical-algorithm statutory subject matter; the test is intended to be consistent witb legislative history and case-law. For examination purposes, "mathematical algorithms" are considered to refer to "methods of calculation, mathematical formulas, and mathematical procedures generally," and no distinction is made between man-made mathematical algorithms and mathematical algorithms representing discoveries of scientific principles and laws of nature (which have never been statutory subject matter). For a process claim involving a mathematical algorithm to be patentable, the claim excluding the algorithm is required to be statutory subject matter-i.e., the claim must be for a process, machine, etc. Trivial post-solution activity like displaying a number is not sufficient. (Patentable Subject Matter: Mathematical Algorithms and Computer ProgramsJ' 1106 0.G.4, Sept. 5, 1989; also contained in *Patent Protection for Computer Software: The New Safeguard,* Michael S. Keplinger and Ronald S. Laurie, (eds.) (Englewood Cliffs, NJ: Prentice Hall Law and Business, 1989), pp. 9-42.)

[23]35 U.S.C. Section 103. See Irving Kayton, *Kayton on Patents* (Washington, DC: Patent Resources Institute, Inc., 1983), ch. 5.

[24]PTO categorizes patents by some 350 classes, each with some 350 subclasses; classification is done according to structural elements. Many software-related patents are classified in classes 364,235, and 340, but not all patents in these classes are software-related. PTO places patents drawn solely to computer processes that are not classifiable in other areas of technology in Class 340, Subclass 300. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18,1989, p. 4)

[25]A common definition of the term *algorithm is:* "a set of ruks which specify a sequence of actions to be taken to solve a problem [or carry out a process]. Each rule is precisely and unambiguously defined so that in principle it can be carried out by machine." (*Chambers* Science and Technology Dictionary, Peter M. B. Walker (cd) (New York, NY: W & R Chambers, Ltd., 1988), p. 23.)

interpretations have been inexact or at odds with common use of the terms by mathematicians, computer scientists and programmers. Thus, legal battles have included arguments over distinctions between "mathematical" and "nonmathematical" algorithms, mathematical algorithms and "numerical" equations, equations and "laws of nature" or "basic truths," algorithms and "mental steps," etc.[26]

While algorithms may be numerical or non-numerical, algorithms are all "mathematical" constructs. Therefore, making distinctions between mathematical and non-mathematical algorithms, or even between algorithms and computer programs, is problematic for the long term.[27] Moreover, while a particular algorithm may describe a new or improved method of carrying out an operation (like a Fourier transform), or even the most computationally efficient, it may not describe the only method.

Trade Secret

Trade secret law protects the owners of certain information against its misappropriation. Others, who have not obtained the information by improper means, are free to use the information and associated ideas. Unlike copyright or patent, there is no limitation on its duration. Trade secret has been the traditional favorite form of protection for mainframe and minicomputer software. From the viewpoint of a software developer, the advantages to trade secret are that it protects a program's underlying ideas, logic, and structure, not just expression (as in copyright). It avoids formalities of registration or application and lengthy waits for protection. Enforcement is relatively clear-cut and injunctions or compensatory relief is available for those who can prove misappropriation of trade secrets.

On the other hand, trade secret protection doesn't protect against independent creation, reverse engineering, or accidental disclosure of the secret. Also, it can be costly or impossible to maintain secrecy, and the lack of uniformity in State and national laws can be frustrating. In the United States, trade secrets are protected by individual State laws, although there is a Uniform Trade Secrets Act enacted in many States with minor variations. (Although most developed nations have some form of protection for confidential business information, most foreign nations outside of Western Europe do not have trade secret laws per se.) However, much of what trade secret law does can be accomplished by contract and by enforcing licensing terms against disclosure.

When software is protected by trade secret it maintains that status so long as it is not publicly disclosed. For society, this can lead to a lack of knowledge about the state-of-the-art.[x] In turn, this can adversely affect prior art for patent examinations and lead to "reinventing the wheel" rather than building on (or around) prior advances.

Maintaining Software as a Trade Secret

For software (or anything) to be protected as a trade secret, it must not be generally known to a competitor, there must bean effort to maintain its "secrecy," and **those** to whom the secret is disclosed must have a duty not to mistreat the information. However, if they do and a third party gets hold of it, then in many jurisdictions, the third party has no duty to respect the trade secret.

Trade secret protection became popular long before the wide proliferation of personal computers (PCs). Markets (for mainframe and minicomputer software) were smaller then, and much software was custom-developed for particular clients or small market niches.

Some parts of the software market still work like this. In these types of markets, trade secret software has relatively limited exposure, usually to users with contractual obligations to the developer. Often, software is delivered to the client in a lower-level language like machine code (sometimes called object code), with contractual agreements prohibiting the client from reverse-compiling it to yield equivalent code in a more easily analyzed, higher-level language like Fortran.[29]

---

[26] For instance, case law has sometimes treated the term "mathematical algorithm" as synonymous with a "mathematical formula" such ss sin(2a) = 2(sin(a)cos(a)) or a "law of nature" such as E = (mass) (speed of light squared). (For discussions see: Donald S. Chisum, "The Patentability of Algorithms," *University of Pittsburgh Law Review, vol. 47, No. 4,* summer 1986, pp. 959-992; and Supreme Court cases *Gottschalk v. Benson, 409* U.S. 63 (1972), *Parker v. Flook,* 437 U.S. 584 (1978), and *Diamond* v. *Diehr, 450* U.S. 381 (1981).)

[27] For a computer scientist's perspective on legal confusions resulting from unsuitable models for algorithms and computer programs, see Allen Newell, "The Models Are Broken, The Models Are Broken!" *University of Pittsburgh Law Review, vol. 47, No. 4,* summer 1986, pp. 1023-1035.

Some think that new and useful algorithms, including "mathematical" algorithms should constitute subject matter eligible for patent protection (see Donald S. Chisum, ibid., pp. 959-1022.)

A recent decision by the Court of Appeals for the Federal Circuit (*In re Iwahashi, et al.,* CAFC 89-1019, decided Nov. 7, 1989) reversed PTO's rejection of a patent application in which the algorithm constituted the bulk of the invention. Some observers consider that this decision, which limited patent protection for the algorithm only to ita use in the particular apparatus described in the claims, will further ease the way for patent protection for algorithms. (See Edmund L. Andrews, "Patents: Algorithm Ruling May Aid Software," *The New York* Times, Nov. 11,1989, business section, p. 36.)

[28] By contrast, patent rights are granted in exchange for full disclosure of the patentee's "beat method" of practicing the invention. But some consider that in practice claims are sometimes so broad that they don't really show the state-of-the-art (B. Kahin, personal communication, Dec. 1, 1989),

[29] Absent contractual agreements, trade secret does not protect against reverse engineering.

[30] The Copyright Office has special release provisions for deposit of software with trade secrets. Depositors may use object code, remove trade-secret pare% etc.

PC software, by contrast is mass-marketed to hundreds of thousands of customers.[30] To help maintain trade-secret status, PC software is often published in object code and distributed with a "shrink-wrap" license, which every purchaser is supposed to agree to on opening the package. The shrink-wrap license may contain language and terms that purport to create a duty by the user to maintain the trade-secret information, but some question whether this would stand up in court.[31]

[30] The Copyright Office has special release provisions for deposit of software with trade secrets. Depositors may use object code, remove trade-secret parts, etc.

[31] See Anne W. Branscomb, "Who Owns Creativity?" *Technology Review*, May/June 1988, p. 43., and also Sherman et al., op. cit., footnote 2, sec. 309.4(g). Anne Branscomb believes that statutory clarification is needed for the status of trade secrets within copyrighted works (Anne W. Branscomb, personal communication, Dec. 8, 1989).

Appendix B
International Protection for Computer Software

Intellectual-property issues are of growing international **concern.** Problems like commercial piracy[1] that occur in domestic markets have international counterparts. The United States currently enjoys a strong competitive position in international software markets, and appropriate intellectual-property protections and enforcement can help maintain our position and reduce piracy.

With an emphasis on software protections, this appendix briefly reviews existing multilateral and bilateral treaties that help protect the intellectual property of a U.S. national via copyright and patent.[2]

international *Conventions and Treaties*

Copyright is the predominant form of software protection in the United States and abroad. In most countries, computer programs per se are not in principle eligible for patent protection (although interpretations of these policies vary in practice among the various patent offices and courts). However, in some countries (including the United States) certain types of computer-implemented processes and algorithms can be patented.[3]

Copyright and patent protections abroad are substantially similar in form to those in the United States, and have most of the same advantages and liabilities. Sui generis protection for software has been proposed but has not had much of an international impact thus far.[4]

Copyright

Copyright protection abroad is provided for U.S. nationals principally through the Berne Convention and the Universal Copyright Convention.[5] The United States formally joined the Berne Convention in March 1989. The treaty was first established in 1886 and is the primary multilateral agreement in the world dealing with copyright. It is administered by the World Intellectual Property Organization (WIPO), an agency of the United Nations. Berne's fundamental principle of "national treatment" requires each member nation to provide the same protection to works of nationals of other member nations as it does to works of its own nationals. Berne requires that a nation provide certain minimum rights in order to join the convention, including moral rights for the author[6] and automatic protection, thus eliminating the former

---

[1] "Piracy" has been defined as "the reproduction and sale of copyright material without the consent of author or publisher," by Publishers Association and the International Federation of Phonogram and Videogram Producers on Behalf of the U.K. Anti-Piracy Group, 1986, cited in Mark L. Damschroeder, "Intellectual Property Rights and the GATT: United States Goals in the Uruguay Round," *Vanderbilt Journal of Transnational Law*, vol. 22, No. 2,1988, p. 368, footnote 1.

In this paper, OTA uses the term "piracy"to mean unauthorized commercial reproduction and sale, not unauthorized private (noncommercial) copying. (For a discussion of the legal status of private copying, see U.S. Congress, Office of Technology *Assessment, Copyright andHome Copying: Technology Challenges the Law*, OTA-CIT-422 (Washington, DC: U.S. Government Printing Office, October 1989), ch. 3.)

[2] Only a few countries have extensive trade secret laws.

[3] The Supreme Court has not ruled on whether computer programs per se are patentable subject matter, but has ruled that computer-implemented algorithms that are deemed "mathematical algorithms" per se are not statutory subject matter. Courts have thus held that a computer process or algorithm is statutory subject matter unless it falls within a judicially determined exception like the one for "mathematical algorithms" per se. U.S. Patent and Trademark Office (PTO) examiners use atwo-parttestto decide whether patent claims containing ''mathematical algorithms" are statutory subject matter. (U.S. Patent and Trademark Office, "Patentable Subject Matter: Mathematical Algorithms and Computer Programs," 1106 O.G. 4, Sept. 5, 1989).

Inthispaper, OTAsometimes uses phrases like "patents for software-related inventions;""software-related patents," or "patenting algorithms" to refer generally to patent protection for computer-implemented processes and algorithms. ThePTO considers terms like "software patents" to be a misnomer because they may be interpreted to mean that a computer program per se (i.e., the sequence of coded instructions itself) is patentable, as opposed to the underlying computer process it carries out. (M. Keplinger, G. Goldberg, and L. Skillington, PTO, comments on draft paper, Dec. 18, 1989, pp. 1-2.)

[4] The World Intellectual Property Organization (WIPO) proposed sui generis Model Provisions on the Protection of Computer Software providing a mixture of patent and copyright protection. The model provisions were not based on the principle of national protection, instead giving computer software explicit and absolute rights and protection in all signatory nations. Intended as a guideline for national legislatures, the model provisions have not been adopted.

[5] While copyright tends to extend similar protection in most countries, there are national differences. In West Germany, for example, computers are considered a functional work and must meet a relatively high standard of "originality." One estimate suggests that 90 percent of programs will fail to meet that standard. (From the *Inkasso* case, cited by Ian A. Staines, "An Assessment of the European Commission's Proposal for a Council Directive on the Legal Protection of Computer programs," *The Computer Lawyer*, vol. 6, No. 9, September 1989, p. 21.)

[6] Moral rights "are separate from the economic rights of the author and concern what are usually called rights of paternity and integrity. The right of paternity is* right to be named as author of a work; the right of integrity is the right to object to distortion, other alteration of a work, or derogatory action prejudicial to the author's honor or reputation in relation to a work. Article 6*bis* of the [Berne] convention provides that authors shall have the rights of paternity and integrity. Congress concluded that present law, including unfair competition law and State and common law protection [such as libel, defamation, or misrepresentation], provides sufficient moral rights to fulfill the obligations of the Berne Convention. Therefore, it is not necessary for the implementation act to include additional moral rights." (U.S. Copyright Office, Circular 93a: The United States Joins theBerne Union, February 1989, p. 3.)

Under the Constitution, the United States does not accept a "natural right" theory of copyright giving inherent moral rights to the fruits of one's own labor. The United States has historically considered economic incentives for creativity as the basis for copyright protection.

U.S. requirements for formal notice and registration.[7] The latter two provisions were perceived as substantial barriers to entry by the United States.

The United States is also a member of the Universal Copyright Convention (UCC), which was established and adopted by the United States in 1955. It is administered by the United Nations Educational, Scientific, and Cultural organization (UNESCO), an agency of the United Nations. The United States withdrew from UNESCO in 1984 but adheres to the Convention.[8] UCC is also based on the principle of national treatment but it provides less protection than the Berne Convention and has lower minimum standards. In nations that agree to both Berne and UCC, Berne takes precedence.

The Berne Convention is recognized in 79 nations, which gives U.S. nationals protection in 24 countries where there was no previous copyright agreement.[9] The United States has bilateral copyright agreements with 33 nations as well, often in addition to common Berne or UCC membership.[10] Japan, the members of the European Community (EC), Australia and Canada are members of both conventions, while the Soviet Union is a member of the UCC only.[11] While this leaves a large number of nations in which U.S. works are not protected, the geographic scope of copyright protection is broad. The procedures are simple: once copyright exists for a work in a member nation, it applies in all other signatory nations, according to their own laws. Computer programs are not specifically mentioned in either convention, but are commonly agreed to be included.[12]

Patent

Securing patent protection in foreign countries is a much more difficult process than obtaining a copyright. Patents for any invention are difficult to obtain, due to the rigorous standards of novelty and nonobviousness. A patent must be applied for in each country where it is to be valid-there is no universal patent process.[13] This results in expenditures of time, money, and expert help needed for dealing with differences in languages and requirements.

In most countries software per se is not considered patentable. In many countries (including the United States) patents can be obtained for computer-implemented processes and algorithms (see footnote 3). In some nations (including Canada the USSR, and members of the European Economic community) a patent will not be granted if the novel step is the computer program itself, although in these countries merely having a computer program as part of the invention need not automatically disqualify it from patent consideration.[14]

The United States is a member of the oldest and most extensive patent treaty, the Paris Convention established in 1883. This Convention is based on "national treatment," where both domestics and foreigners are accorded the same treatment. However, there is no requirement that software-related inventions be considered patentable.

Other conventions exist that make international patent protection more convenient, although still not easy. Through the European Patent Convention (EPC), a single application for a patent is valid in up to 11 Western European member nations. The patentee must pay an extra fee for each country included, but only goes through a single application and examination. The EPC does not, however, provide uniform protection; a patent is subject to the existing laws in each of the member countries. A second convention is the Patent Cooperation Treaty, which provides for: 1) an international search report for prior art, 2) a preliminary examination report for some countries, and 3) the option to delay applying for a foreign

[7]The Berne Convention Implementation Act has repealed the mandatory copyright notice requirement (the encircled "c," date, and name of the copyright owner) and eliminated the requirement to register a work at the Copyright Office. Although foreign authors need not register, there are significant incentives for a U.S. citizen to register because for them registration is a precondition for a copyright lawsuit, award for attorney's feea, and statutory damages. (U.S. Copyright (Mice, Circular 93a, op. cit., footnote 6, p. 3.)

[8]The United States is active on an intergovernmental committee for the UCC and also contributes to and supports the copyright-related activities of UNESCO.

[9]Figures from the U.S. Copyright *Office, Circular* 93: Highlights of U.S. Adherence to the Berne Convention, April 1989, and Circular 93a, op cit., footnote 6. Additional members may have joined since these circulars were published.

[10]U.S. Copyright Office, Circular 38a: International Copyright Relations of the United States, July 1989.

[11]Ibid.

[12]See, for example, Ingrid M. Arckens, "Obtaining International Copyright Protection for software: National Laws and International Copyright Conventions," *Federal Communications Law Journal,* vol. 38, August 1986, pp. 283-300. Max W. Laun, "Improving the International Framework for the Protection of Computer Software," *University of Pittsburgh Law Review, vol. 48,* summer 1987, pp. 1151-1184.

There are questions, however, about exactly what protection extends to (i.e. is "look and feel" protected by copyright, what is included in fair use, etc.), similar to the debates currently occurring in the United States over copyright protection for software.

[13]The European Patent Convention, described in more detail later, does provide multinational recognition of a patent. The European Patent office (EPO) makes it somewhat easier to obtain patent rights in Europe.

[14]In Canada, a program is not patentable but an invention involving a computer program would not be rejected outright. In Greece, computer programs per se are unpatentable. In Brazil, computer programs are not patentable, but semiconductor-chip firmware is patentable; in New Zealand computer programs can be patented only indirectly, by patenting hardware programmed in accordance with the program. South Africa has a statutory exclusion for computer programs. In the USSR, patent applications are not accepted for examination if the claimed subject matter is an algorithm or computer program. (Baxter-Sinnott, *World Law and Practice,* vol. 2A (New York, NY: Matthew Bender, 1985), pp. 2A-10 to 2A-12.)

patent for up to 30 months after the initial filing. An applicant must still file in each country (or region such as theEC)separately,but has greater assurance of being

Trade Secret

Trade secret has been the traditionally favored method of protection for mainframe and minicomputer software developers in the United States. However, most countries outside the United States and Western Europe do not recognize either domestic or international trade secret protection, although they may have laws concerning confidential business information that may be similar if less extensive. Japan is developing a trade secret law; however, it is not formulated to protect software.[15] No international conventions for trade secret exist. The validity of trade secret protection for mass-marketed software (commonly used for PC software) is questionable in the United States and there are signs that "shrink-wrap" licensing may not be considered valid in the European Community in the future.[lb]

Bilateral Negotiations

Bilateral agreements are another way to protect intellectual property abroad. In 1984, Congress amended the Trade and Tariff Act of 1974 to require that intellectual-Property protection be considered in awarding benefits under the Generalized System of Preference (GSP) for trading partners.[17] Another clause of the 1974 Act section 301, gave the president the power to restrict imports in retaliation for foreign trade practices that unfairly restrain U.S. trade. This was strengthened in the Omnibus Trade and Competitiveness Act in 1988. The amendment, known as "Special 301," directs the U.S. Trade Representative (USTR) to identify "priority countries" that provide inadequate or ineffective intellectual-property protection. If it is determined that sufficient progress is not being made by these nations, the USTR may bring an unfair trade practice case against the offending country .18 These pieces of legislation attempt to move countries with historically weak protection towards international standards.

For example, the Republic of Korea Singapore, and Taiwan have recently negotiated intellectual-property agreements with the United States. The United States began bilateral negotiations with each nation in the early 1980s. Then the United States began to apply trade leverage around 1985, often through the GSP system or section 301 of the Trade Act. Maintaining relations with the United States is important to each of these countries for economic and security reasons. Singapore and Taiwan, as emerging centers of high technology in the Region, will benefit from stronger protection laws. The laws of all three nations protect software expressly under copyright, and to the extent established by international standards, although only Korea has joined the Paris Convention and the UCC.[19]

Trade and Competitiveness Issues in the Global Economy

If the reasons for domestic intellectual property protection are principally economic, the same is true for international protection. Software protection has both direct and indirect effects on trade and competitiveness. Commercial piracy and loss of royalties result in direct revenue losses to U.S. firms and the U.S. economy. Appropriate intellectual-property protection can encourage investment and innovation and indirectly strengthen the U.S. economic position in high technology and in the business and manufacturing industries supported by computers and software.[20]

The United States is the world's leading innovator and producer of computer software. Estimates of market shares, volume, and revenues vary, but one European study estimates that the United States supplies 70 percent of the world's software and accounts for half the world demand.[21] Another article claims that IBM, the largest software developer in the world, accounts for 60 percent of volume in world software sales and perhaps 70 percent of world operating profits.[22] Western Europe is estimated to have 10 percent of world sales and the Japanese 15 percent. The Soviet Union contributes practically no sales

[15] Pamela L. Hamilton, "Protections for Software Under U.S. and Japanese Law: A Comparative Analysis," *Boston College International and Comparative Law Review*, vol. 7, summer 1984, p. 390.

[16] Michael D. Scott, "Europe 1992: The Impact of Unification on Non-European Computer Companies," *International Computer Law Advisor*, May 1989, p. 5. See also the section on maintaining software as a trade secret in app. A.

[17] R. Michael Gadbaw and Time@ J. Richards (eds.), *Intellectual Property Rights: Global Consensus, Global Conflict?* (Boulder and London: Westview Press, 1988) especially p. 6; and Robert PBenko, *Protecting Intellectual Property Rights (Washington, DC: American Enterprise Institute, 1987)*, p. 11.

[18] Ann Main, "*Pursuing* U.S. Goals Bilaterally: Intellectual Property and 'Special 301 '," *Business America*, vol. 110, No. 19, Sept. 25, 1989, p. 6.

[19] These particulars were all taken from Gadbaw and Richards, op. cit., footnote 17, chs. 8,9, 10.

[20] "Intellectual property rights promote innovation and intellectual creativity. Their protection and enforcement are essential to the expansion of ternational trade, investment, economic development, and.. the beneficial distribution of technology." ("United States Proposal for Negotiations on %-Related Aspects of intellectual Property Rights," *International Computer Law Advisor*, June 1989, p. 13.)

[21] Commission of the European Communities, "Green Paper on Copyright and the Challenge of Technology--Copyright Issues Requiring Immediate Action," June 1988, pp. 171-172.

[22] Figures cited in Gene Bylinsky, "The ~@ Tech Race: Who's Ahead? *Fortune*, vol. 114, Oct. 13, 1986, p. 28.

to the West, trying instead to catch up with Western state-of-the-art.[23] The United States is also the international leader in electronic databases: two-thirds of all databases available on world markets are U. S.-based.[24] According to one estimate, the total revenues generated for the U.S. computer and data processing industries from foreign markets came to $22 billion in 1987; approximately $8 billion of that was from Software.[25]

International Piracy and the Third World

Commercial piracy results in direct revenue losses to U.S. firms, through loss of sales, loss of royalties, and/or loss of investment opportunities.[26] However, redress of piracy abroad is often difficult and can involve issues of technology transfer and assistance to developing nations. Most of the industrialized, developed countries have strong intellectual-property protections, whereas many of the lesser-developed countries either do not have strong intellectual-property laws or do not enforce them.

Nimmer and Krauthaus[27] give two possible reasons for this lack of enforceable protection. The first is uncertainty about the ambiguous position of software in relation to copyright or patent protection. The United States spent several years deciding whether and how much to protect software in the realms of copyright and patent. More recently, Western Europe and Japan have developed protection schemes for software. In the Third World, where software development itself is much younger, legal solutions to protection may be slower than in the more advanced nations.

The second is a North-South trade and technology transfer issue, with the views of advanced nations in conflict with those of the lesser-developed nations. Advanced nations want to protect the computer and software industries that are strong sectors in their economies and want to promote free trade to benefit from these investments. lesser-developed countries want low-cost access to technology in order to promote and modernize business. Many also want to encourage a fledgling domestic programming industry. The advanced nations argue that strong software protection will encourage both domestic innovation and foreign investment; for some nations this argument may be well received, but for others whose development as a high-technology center is much further in the future, if at all, there is less urgency.

GATT Negotiations

Intellectual property has been included in the current Uruguay Round of GATT negotiations, scheduled to conclude in 1990. The GATT (General Agreement on Tariffs and Trade) is the major document regulating trade in the world, and has not traditionally included intellectual property within its sphere.[28] 'The objective of a GATT intellectual property agreement would be to reduce distortions of and impediments to legitimate trade in goods and services caused by deficient levels of protection and enforcement of intellectual property rights:' the U.S. position states.[29] The European Economic Community, Japan, and several other nations have also submitted proposals in support of including intellectual property issues in the GATT negotiations, and the United States is trying to get support from nations such as South Korea and Singapore that already have developed intellectual-property laws.[30]

If a GATT agreement is reached, the parties would adopt laws with a sufficient amount of intellectual-property Protection-' 'sufficient" to be determined relative to domestic law and international standards. It would cover patents, copyrights, trademarks, trade secrets, and semiconductor mask works.[31] Conciliation and dispute settlement procedures would be invoked when informal meetings fail to settle differences between two nations. Finally, strong enforcement measures would allow border control and the withdrawal of GATT concessions if the terms fail to be honored. Enforcement is a particularly important issue to many U.S. software manufacturers, since currently there is often little they can do and few remedies against foreign infringers.[32]

[23] Ibid.

[24] Clarence J. Brown, 'The Globalization of Information Technologies," *The Washington Quarterly, vol.* 11, winter 1988, p. 94.

[25] U.S. International Trade Commission, "'he Effects of Greater Economic Integration Within the European Community on the United States," July 1989, ch. 4, p. 39.

[26] A fuller breakdown and discussion of the many ways through which companies experience revenue 1oss can be found in a study by the U.S. International Trade Commission, "Foreign Protection of Intellectual Property Rights and the Effect on U.S. Industry and Trade," February 1988, ch. 4.

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

[28] However, in the preceding round of negotiations (the Tokyo Round) an anticounterfeiting code was discussed.

[29] "United States proposal for Negotiations. . .," op. cit., footnote 20, p. *13.*

[30] Gadbaw and Richards, op. cit., footnote 17, chs. 8 and 9.

[31] Text of the proposal submitted by the United States to the GATT, printed in "United States Proposal for Negotiations. . .,'' Op. Cit., footnote 20, pp. 15-16.

[32] See Damschroeder, op. cit., footnote 1; Gadbaw and Richards, op. cit., footnote 17, ch. 2; and Dana Williamson, "Addressing Inadequate Intellectual Property protection in the Uruguay Round," *Business America,* vol. 110, No. *9,* Sept. 25,1989, pp. 4-5.

Innovation and Competition

Continued innovation is of great importance to a high-technology industry such as software. Intellectual-property protection encourages innovation by providing incentives for creation and providing some security for investors. This promotes international competition; if innovation increases domestically, the United States can continue to outshine foreign competitors. The United States is thus far the leading and most innovative software producer in the world. Not only has the industry developed earlier here than in other countries, but there is also a large installed hardware base and domestic market which makes investment less risky. So far, U.S. industry's position in the world market and the individualistic approach and enterprise of start-up companies have kept the United States ahead of all competition.

However, there is fear of foreign competition, especially from the Japanese.[33] The Japanese are funding efforts in programming R&D; they planned to spend $125 billion over 10 years, according to one estimate.[34] Artificial intelligence is a major project in Japan; known as the "fifth generation" project, the general goal is to make computers think more like humans. Some U.S. researchers fear that Japanese experience in this area could give them a head start in parallel processing and other cutting edge programming techniques.

[33]For example, Charles Ferguson at MIT points out how the Japanese *took control* of the initially American semiconductor industry, and are now the world leaders. (Thomas Kiley, "High Tech Heresy," *New England Business*, vol. 10, November *1988*, pp. 62-66.) Also see Dewey, Ballentine, Bushby, Palmer, and Wood, "Japanese *Software: The* Next Competitive Challenge," prepared for ADAPSO, January *1989*.

[34]Brown, op. cit., footnote 24, p. 90.