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 software-*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-imp l emented 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?