1995

The Lack of Protection Afforded Software under the Current Intellectual Property Laws

Himanshu S. Amin

Follow this and additional works at: http://engagedscholarship.csuohio.edu/clevstlrev

Part of the Computer Law Commons, and the Intellectual Property Law Commons

How does access to this work benefit you? Let us know!

Recommended Citation


This Article is brought to you for free and open access by the Law Journals at EngagedScholarship@CSU. It has been accepted for inclusion in Cleveland State Law Review by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.
THE LACK OF PROTECTION AFFORDED SOFTWARE UNDER THE CURRENT INTELLECTUAL PROPERTY LAWS

I. INTRODUCTION .................................................. 19
   A. Software Piracy ............................................. 19
II. SOFTWARE BASICS .................................................. 20
   A. Transistors ................................................... 20
   B. Source and Object Code .................................. 21
III. METHODS OF PROTECTING SOFTWARE ......................... 21
   A. Patents ....................................................... 22
   B. Copyrights ................................................... 30
   C. Trade Secrets ............................................... 38
IV. INCOMPATIBILITY OF THE VARIOUS FORMS OF PROTECTION .... 38
V. CONCLUSION ...................................................... 40

I. INTRODUCTION

Many abstract advances in computer technology remain unprotected since the current intellectual property system has been shaped through a focus on tangible, physical inventions. The software industry in the United States "accounts for domestic revenues of over fifty billion dollars each year in worldwide sales and services." Accordingly, it is imperative that United States software developers be provided adequate intellectual property coverage in order to protect existing technology and encourage further innovation in the field. The present lack of adequate protection has handicapped American developers unnecessarily in the global software market.

A. Software Piracy

As described by some commentators, "Software piracy in the [United States] has reached epidemic portions. For every legitimate copy of a commercially
successful program at least four, and according to some estimates ten or more, pirated copies exist.5 Unauthorized copying of software by private and corporate consumers has resulted in the software industry losing over a billion dollars annually in the United States alone. In addition to the billions lost annually as a result of domestic piracy, international piracy costs the software industry between $8 billion and $10 billion each year in lost revenues.6 Software publishers are thus deprived of revenues that could be used to develop new products and support existing software.7 "At present, software development may account for fifty to ninety percent of the total development costs of a typical industrial or scientific system..."8 Piracy also makes it difficult for new software publishers to attract venture capital. "This lack of money for research and development could weaken the software industry—an industry in which the constantly advancing state of the art makes financial support for innovation essential."9

II. SOFTWARE BASICS

For the reader to understand the issues discussed in this paper, it is necessary to explain a few basic concepts relating to the technology behind computer software.

A. Transistors

A personal computer's microprocessor is a complex collection of thousands or even millions of microscopic transistors which serve as on/off switches. The transistors are laid out in a microchip along circuits made up of superfine traces of aluminum. When the transistors are arranged in certain patterns, parts of the microprocessors are designated to hold data while others are used to manipulate that data.10

The on/off switches of the transistors lend themselves handily to representing binary numbers. Data and software codes are stored in the form of binary numbers. In the binary system, two digits—0 and 1—represent all


7 See Julie A. Mark, Software Copying Policies: The Next Step in Piracy Prevention, 2 J.L. & TECH. 43 (1987) (research indicates that publishers now devote from 5% to 10% of their revenues to research and development).

8 Phillips, supra note 4, at 1001 (citing BARRY W. BOEHM, SOFTWARE ENGINEERING ECONOMICS 18 (1981)).

9 BOEHM, supra note 8, at 18.

10 See RON WHITE, HOW SOFTWARE WORKS 5 (1993).
LACK OF PROTECTION AFFORDED SOFTWARE

numbers. A transistor that is turned off represents a 0; a transistor that is turned on represents a 1. Each single 0 or 1 is referred to as a bit; eight bits make up a byte, and 1,024 bytes make up a kilobyte (K).11

B. Source and Object Code

Computer programs most often exist in two forms: source code and object code. Source code is the language in which software is actually written by the programmer,12 it is a high-level form of language which resembles our spoken language.13 Source code is programming language that is intelligible to humans, and it is comprised mainly of descriptive key words and common mathematical notations.14 Source code is usually translated by a compiler program into object code, which is generally only readable by the computer.

Object code is considered to be low-level language, and it is comprised of a sequence of binary ones and zeros that is unintelligible to humans. Object code "flips switches" on and off within the computer.15 It is the object code which is purchased and run in a typical computer program; usually source code is not publicly sold or distributed. It should be noted that translating from object code to source code is very difficult, even with the assistance of a decompiler; it is easier to rewrite a program than it is to translate object code back into source code.16

III. METHODS OF PROTECTING SOFTWARE

There are three main categories of legal protection available for computer software: patent, trade secret and copyright. Patents would seem appropriate as a form of legal protection for computer software since they are designed to protect inventive technical solutions, which is descriptive of computer programs.17 Title 35 U.S.C. § 154, which provides for patent legal protection, protects the functional aspects of an innovation—the inventive idea itself rather than its manner of expression.18 Patent law, however, has historically protected only tangible inventions.19

11Id.


13See WHITE, supra note 10, at 44.


15Id.

16See Taylor, supra note 12, at 192.


18See 35 U.S.C. § 154 (1995) ("Every patent . . . grant[s] to the patentee . . . for the term of twenty years from the date on which the application was filed in the United States . . . the right to exclude others from making, using, or selling the invention . . . ").
A. Patents

Patent protection of computer software has generally been deemed insufficient for numerous reasons. First, it is very difficult to obtain relief in litigation. Second, many types of programmable processes and programmed machines may not ultimately be patentable subject matter. Further there are commercial uncertainties arising from this fact, even should they eventually be held patentable. Fourth, the most valuable programs will not qualify for patents, even if the subject matter is patentable, since they will run afoul of the novelty and non-obviousness requirements. Even if a computer program can be patented, this may be possible only if it is claimed in an awkward and unnatural manner.

Additionally, obtaining a patent on a program is very expensive; it is estimated that the expense associated with obtaining a software patent exceeds $10,000. The above problems exist because a person seeking a patent must first prepare a patent application that designates any claims to the scope of the invention. Then, the application must be filed with the United States Patent and Trademark Office [hereinafter PTO]. The PTO must then examine the application and conduct a search for any prior patents which may limit the scope of the invention. The tremendous number of patents which have been issued makes this process lengthy, and very costly.

19 See Computer Intellectual Property and Conceptual Severance, supra note 2, at 1047.


21 Bender & Barkume, supra note 20, at 280-83.

22 Id.

23 Id.

24 Id. The P.T.O. examines each application to determine whether the claimed invention is novel, non-obvious and useful. If it finds that the invention is, it issues a patent. In determining novelty, the occurrence of certain events must be researched such as the occurrence of a printed publication, the patenting of an invention, public knowledge or use by others, upon which the claims read, prior to the applicant's having invented the invention being claimed, then no valid patent may issue. See, e.g., 35 U.S.C. § 102 (1995). The § 102 requirement for novelty is but a preliminary threshold to overcome. Having done so does not mean that the applicant's claims are patentable relative to the prior art. Thus, even though the applicant has established that his claimed invention is novel (i.e., differs structurally and/or functionally from the prior art that is the claims do not read literally on a single item of the prior art), 35 U.S.C. § 103 (1995) superimposes the requirement that the claimed invention as a whole would also have been nonobvious "at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains." 35 U.S.C. § 103 (1995). See generally, IRVING KAYTON, PATENT PRACTICE 1 (5th ed. 1993).

25 Id.

26 See Szabo, supra note 20, at 526.
Another pitfall is that patent protection may not suit an owner seeking to maintain a competitive edge through secrecy because the patent itself contains an elaborate description of the software and the manner and process of making and using it. The description must be full, concise, and contain exact terms which enables any person skilled in the art of computer software to duplicate it. The patent becomes publicly available after issuance, and any secrecy surrounding the disclosed invention is forfeited because a competitor can examine the patented software and design around it.

Also problematic is that United States patent protection only begins on the date a patent issues from the PTO. A significant interval of time passes between the filing of a patent application and the ultimate issue of the patent. The average period of pendency between filing an application and issue of a patent is eighteen months overall, and for software patents is at least two years. Since most software generally has a short product life, the patent becomes useless because the commercial lifetime of the software passes before a patent would be issued. A final problem is that since the patent itself is a public document, it is difficult to police unlawful use.

The expense of a patent is just one of many hurdles—patent procurement is doubtful in the majority of cases. Patents must be susceptible to industrial application, in other words, must have some tangible character. Algorithms,


30 See Bender & Barkume, supra note 20, at 281.

31 Id.

32 See generally David Bender, Protecting Computer Trade Secrets, at 713, 723 (PLI Patents, Copyrights, Trademarks, and Literary Property Course Handbook Series 1986), available in WESTLAW, 224 PLI/Pat 713.

33 The majority of patent applications do not result in issued patents. Obtaining a patent is initiated by filing a patent application in the Patent and Trademark Office (PTO). The application must comply with various statutory requirements and with rules established by the PTO pursuant to the patent statute (35 U.S.C.). Subsequent to the filing of the requisite application, it is studied and a search is conducted through all the relevant prior United States patents by the examiners in the PTO and also through patents of foreign countries and publications to find out if the invention is new and nonobvious to a person of ordinary skill in the art to which the invention pertains (i.e., in the subject technology). A decision is then reached by the examiner (which is most often negative) in light of the study and the results of the search, as to the patentability of the invention as claimed and also as to various formal matters. See, IRVING KAYTON, PATENT PRACTICE 1 (5th ed. 1993).

34 See Jean Jonqueres, The Patentability of Software, 18 INT'L REV. INDUS. PROP. & COPYRIGHT L. 607, 609 (1987) (stating that patent law generally protects objects which are capable of being handled and operated; it does not protect ideas so the invention must be in the form of a product and not of a concept); Dawn Jordan, Software Piracy:
unless tied to a physical process, are not protected. Software and algorithms have not received the same level of protection that computer hardware has because of their intangible characteristics.

The essence of software does not lie in a process, machine, manufactured article, or composition of matter, but rather in its underlying algorithm. An algorithm is a sequence of steps which, when followed, perform a useful and intended result. An algorithm is meant to be implemented on high-speed processors, but humans, in theory and given enough time, can also perform the algorithm to yield the same result.

The Supreme Court in *Gottschalk v. Benson*, emphasizing that the algorithm was not a tangible process, construed the patent law to exclude protection of an algorithm that converted binary-coded decimal numbers into pure binary numbers. The Court also noted that mental steps are not patentable. The Court reasoned that the patenting of an algorithm would be the equivalent of patenting mental steps, which would result in a patent being able to exclude individuals from thinking in the manner of the patented algorithm. The Court defined an algorithm as "[a] procedure for solving a given type of mathematical problem."

This definition of the term algorithm has resulted in much confusion because it differs from the definition generally used in the computer industry. In the computer industry the term "algorithm" generally means "a procedure consisting of a sequence of logical operations which combine data, mathematical principals and equipment for the purpose of interpreting and/or acting upon certain data."

It must be emphasized that when the Supreme Court in *Benson* used the term "algorithm", it was referring to a procedure for solving a mathematical problem.

---

35 See Chisum, *supra* note 2, at 960-961 & 961 n. 3.

36 *Computer Intellectual Property and Conceptual Severance, supra* note 2, at 1049. It has been noted that, "Patent law's tangibility requirement has its roots in the nineteenth century." *Id.* at 1049 n.24 (citing Clark Thread Co. v. Willimantic Linen Co., 140 U.S. 481, 489 (1891) ("A conception of the mind is not an invention until represented in some physical form . . . ."); see also 35 U.S.C. § 101, note 38 (West 1984) (Physical Form As Essential) (citing numerous cases that require tangibility for patentability)).


38 409 U.S. 63 (1972).

39 See *id.* at 68 (stressing the abstract nature of this patent claim).

40 *Id.* at 67.

41 *Id.* at 65.

and not a procedure for interpreting or acting on input data. Even though the Court limited its holding to the particular facts presented in Benson, the Court concluded that algorithms and software should be excluded from patent protection under the current statute. This does not render all inventions that involve a mathematical formula unpatentable. The holding simply means that a process claim which contains a mathematical formula cannot be drawn so broadly that it covers all uses, known and unknown, of the formula.

The ruling in Benson, rendered algorithms unprotectable when taken by themselves. One commentator has noted that this has caused "entire fields of algorithmic development [to] remain vulnerable to piracy, which may discourage investment and work in the algorithmic field." As an example, this commentator pointed to Professor Ronald Bracewell's work involving the Fast Fourier Transform:

... Professor Ronald Bracewell developed an improvement to the Fast Fourier Transform (FFT), a highly sophisticated algorithm fundamental to the field of signal processing and crucial to many scientific applications. His improvement can provide as much as a twofold increase in the speed of certain calculations, 'regardless of the kind of computer used.'

Professor Bracewell was forced to prototype his algorithm within a silicon microchip, a physical form, in order to satisfy the tangibility requirements of the current patent law.

---

43 Id.

44 See Computer Intellectual Property and Conceptual Severance, supra note 2, at 1050 n.28 (but "Benson does not preclude the patentability of all abstract processes or software programs").

45 See Gottschalk v. Benson, 409 U.S. 63, 72 (1972) (stating that a grant of patent protection to this claim would effectively extend patent protection to an algorithm, which the Court viewed as tantamount to patenting an idea itself).

46 See Jensen, supra note 42, at 1437 n.142; Benson, 409 U.S. at 65-67.

47 See Jensen, supra note 42, at 1455 n.143.

48 See Benson, 409 U.S. at 72.

49 Computer Intellectual Property and Conceptual Severance, supra note 2, at 1058; see also Chisum, supra note 2, at 1020 ("Policy considerations indicate that patent protection is as appropriate for mathematical algorithms that are useful in computer programming as for other technological innovations.... The absence of a clear rule on the allowability of patent claims to algorithms that comply with the rigorous standards of patentability and other requirements for obtaining patent protection may cause reluctance on the part of financial interests to back new ventures for the development of innovative software. . .").

50 Professor of Electrical Engineering at Stanford University.


52 Id.
The patent law’s legal requirement of tangibility imposes unnecessary transaction costs such as seeking a lab to prototype the algorithm in silicon, and it increases product development costs. In addition, the tangibility requirement encourages attempts to "shoehorn claims" into a rigid physical framework. Professor Bracewell finally received a patent under the description "Computer and Method for the Discrete Bracewell Transform." The professor was forced to add the words "computer and" in order to satisfy the patent law’s requirement of physicality.

The gap between intellectual property law and the nature of modern inventions, such as Professor Bracewell’s transform, results in a tremendous waste of resources. The example of Professor Bracewell clearly illustrates the misalignment that distorts all of computer law by the requirement of physicality. "Rather than focusing on the essence of the innovation, the [present patent] law draws an arbitrary distinction between the tangible and the abstract." Current patent law promotes high transaction costs while providing uncertain protection, and leaves "abstract innovations either completely unprotected or distorted and ‘shoehorned’ into some tangible expression."

A formula in the abstract may not be patented; the application of a newly discovered formula in a particular process, however may be patented if the whole process is patentable. When a patent claim recites a formula, the claim must be examined to determine whether it is seeking patent protection for the formula in the abstract or for the whole structure or process applying the formula. The examination ensures the patent laws function to protect only

53 Id.
54 Id.; see Loretto v. Teleprompter Manhattan CATV Corp., 458 U.S. 419, 451 (1982) (Blackmun, J., dissenting) (arguing that the majority’s holding that any permanent physical occupation is a per se taking "encourages litigants to manipulate their factual allegations to gain the benefit of its per se rule"); Parker v. Flook, 437 U.S. 584, 590 (1978) (rejecting an attempt to shoehorn an otherwise unworthy patent claim into a physical process in order to obtain protection).

57 Id. at 1060.
58 Id.
59 Id.
60 Id.
61 See Jensen, supra note 42, at 1437 n.145.
62 Id. at 1437 n.146.
the whole structure or process applying the formula.63 Such a patent claim and the inquiry it involves were presented in Diamond v. Diehr.64 The Diamond Court held that patent laws could afford protection to software that is linked to a physical process.65 The patent application claim satisfies the requirements of 35 U.S.C. § 101 when the computer software is linked to a structure or process applying the algorithms, which when considered as a whole, perform a function that the patent laws were intended to protect.66 

In 1992, the U.S. Court of Appeals for the Federal Circuit in Arrhythmia Research Technology, Inc. v. Corazonix Corp.,67 employing the aforementioned Diehr test (which is really a modified Freeman-Walter-Abele test),68 affirmed a

63Id.

64450 U.S. 175 (1981). Diehr applied for a patent on a process for molding rubber which ensured that the rubber would always be perfectly cured. Id. at 177. A computer operated by a program took constant measurements of the temperature of the mold and constantly recalculated the remaining cure time, by applying a known formula, taking into account changes in room temperature during the elapsed cure time. Id. at 177-78. When the computer determined that the remaining cure time was exactly zero, it automatically opened the mold. Id. Diehr did not attempt to claim patent protection for the formula itself; he claimed only the improved process for curing rubber. Id. at 187. Since Diehr only attempted to patent a total process and not the formula itself or a method of calculation, the Court held that the presence of the formula and the computer solution of it did not destroy the statutory subject matter of the process as a whole. Id.

65450 U.S. at 187. The Court explained that patentability will depend on whether the subject matter is a mathematical formula or an industrial process. Id.

66See Diehr, 450 U.S. at 192. Since Diehr, the lower courts have followed a two-step analysis in deciding whether patent claims for computer related inventions, which are usually predicated upon an assertion that the application of a formula is part of a particular process, involve one of the categories of statutory subject matter or one of the exclusions to those categories. This analysis was originally set forth in In re Walter, 618 F.2d 758 (C.C.P.A. 1980), and In re Abele, 684 F.2d 902 (C.C.P.A. 1982), and was approved by the Supreme Court in Diehr. The claim is first analyzed to determine whether a mathematical algorithm is directly or indirectly recited. See Jensen, supra note 42, at 1440. Next, if a mathematical algorithm is found, the claim as a whole is further analyzed to determine whether the algorithm is "applied in any manner to physical elements or process steps," and, if it is, it "passes muster under 35 U.S.C. § 101." Id.

67958 F.2d 1053, 1058-1060 (Fed. Cir. 1992).

68The test poses two inquiries. First, does the patent claim recite, directly or indirectly, a mathematical algorithm, formula or "mental step"? If not, then the claim is for statutory subject matter, if so, the second inquiry must be reached. Second, does the claim involve application of the algorithm, etc. to specific physical elements or processes (something more than a field-of-use limitation or the addition of "non-essential post-solution activity"). If so, the claim is for statutory subject matter; if not, the claim is not for statutory subject matter.

The second step's content has expanded and contracted over time. In In re Freeman, 573 F.2d 1237, 197 U.S.P.Q. 464 (C.C.P.A. 1978), the CCPA framed it in terms of algorithm "preemption": i.e., "the claim must be . . . analyzed to ascertain whether in its entirety it wholly "preempts." In In re Walter, 618 F.2d 758, 205 USPQ 397 (CCPA 1980), the court reframed it as follows:
district court determination of patent infringement and validity on a patented apparatus and process for human heart electrocardiographic signal analysis. 69 The Federal Circuit upheld patentability, even though the claims included a mathematical algorithm, "because the output of the system was 'not an abstract number but [was] a signal related to the patient's heart activity.'" 70 The Arrhythmia case suggests that as long as a "computer program produces external output, rather than just performing endless internal calculations without practical output, the test for patentability is satisfied." 71 However, very few computer programs would fail the Arrhythmia test. 72 As a result, the U.S. Patent and Trademark Office has been flooded with an onslaught of patent applications drafted accordingly with the Federal Circuit's open-handed approach. 73 The software must now be disguised under the pretext of a process in order to obtain patentability. One clearly foreseeable problem is that many broad process claims that have become patentable also cover underlying software. 74 The PTO will be unduly burdened with construing manufacturing

Once a mathematical algorithm has been found, the claim as a whole . . . must be further analyzed. If it appears that the mathematical algorithm is implemented in a specific manner to define structural relationships between the physical elements of the claim (in apparatus claims) or to refine or limit claim steps (in process claims), the claim being otherwise statutory, the claim passes muster under § 101. If, however, the mathematical algorithm is merely presented and solved by the claimed invention, as was the case in Benson and Flook, and is not applied in any manner to physical elements or process steps, no amount of post-solution activity will render the claim statutory; nor is it saved by a preamble merely reciting the field of use of the mathematical algorithm.

Various indicia are helpful in determining whether a claim as a whole calls merely for the solution of a mathematical algorithm. For instance, if the end product of a claimed invention is a pure number, as in Benson and Flook, the invention is nonstatutory regardless of any post-solution activity which makes it available for use by a person or machine for other purposes. If, however, the claimed invention produces a physical thing, such as the noiseless seismic trace in In re Johnson, [589 F.2d 1070, 200 USPQ 199 (CCPA 1978)] the fact that it is represented in numerical form does not render the claim nonstatutory.

70 Id. (quoting Arrhythmia, 958 F.2d 1053, 1058-60 (Fed. Cir. 1992)).
71 Id.
72 Id.
73 Id.
74 For example, as explained by Roger Cook:
[A] typical patent on a software program for controlling a manufacturing process might read as follows:
A process for controlling the implementation of engineering changes in a semiconductor manufacturing process comprising:

http://engagedscholarship.csuohio.edu/clevstlrev/vol43/iss1/7
LACK OF PROTECTION AFFORDED SOFTWARE

processes as prior art for software patent applications. Many critics believe that the U.S. Patent and Trademark Office does not do a good job of examining software patents. According to this view, the Patent and Trademark Office's prior art library does not include an adequate amount of software prior art.

Software described in a patent specification, can be examined by anyone because algorithms cannot be patented by themselves. Hence, anyone could determine the algorithm employed by a particular software and use the algorithm itself free of any fear of infringing the patent. However, the algorithm can not be used in a process or in an apparatus which, when taken as a whole with the algorithm, would infringe any claim in the patent.

Obtaining patent protection for an apparatus or a process with a software implemented algorithm does not deter would be infringers. Patentees have the burden of detecting infringement on their patent. Even when infringement is suspected by a patent owner, license negotiations between the patent owner and the suspected infringer frequently take several years.

Any person or entity charged with patent infringement can challenge the validity of the patent in federal court. One of the factors that justifies this rule is the acknowledged inability of the PTO to discover all of the relevant prior art during the patent application process.

Professor Paul Goldstein, a professor of intellectual property law at Stanford University's law school, has stated that many of those concerned about Compton's broad-based patent complain that the Patent Office may have been unable to research the sources of the technology which underlie the invention. Victoria Slind-Flor, Rethinking Protection, NAT'L L. J., Jan. 24 at s1, s26. According to Professor Goldstein, "It reflects the inefficacy of the patent office in dealing with software—the lack of a prior art data base to measure software—related inventions against." 

Any patent does not give its owner the right to practice the patented invention, only the right to prevent others from practicing it. Companies with patented technology often discover that others—sometimes numerous others—hold blocking patents that make it impossible or impracticable for the company to use its own inventions.

The small company that wishes to keep its patented software to itself, and not license it to anyone—let alone to one of the major players in the industry—is going to be under terrific pressure to succumb to the
infringer reaps a tremendous windfall from the license negotiations since he is usually not required to pay patent royalties on infringing activities that occurred prior to the time in which the license agreement takes effect.\textsuperscript{81}  

If licensing negotiations fail, litigation is another option for the patent owner to pursue. However, patent litigation can be very complex and may stretch over many years (seven years is a common time-frame in the United States for patent litigation).\textsuperscript{82} During litigation, the patentee runs the risk of losing if a court holds his patent invalid. If the patent is found to be valid and infringed, patent remedies include injunctions\textsuperscript{83} and substantial financial recovery (such as monetary damages, court costs; and if willful infringement is found, the possibility of treble damages and attorneys fees).\textsuperscript{84} Nonetheless, the remedies afforded the patent owner may be of little consolation in light of the uncertain outcome of complex and time consuming litigation. Consequently, the vast majority of patent cases are settled with the infringer, leaving the negotiation table, paying far less than he would have if found liable of infringement.\textsuperscript{85}

**B. Copyrights**

Presently, copyright statutes are the predominant avenue\textsuperscript{86} used to protect intellectual property rights in computer software.\textsuperscript{87} In the Copyright Act of 1976, Congress clearly expressed its intent that computer programs be considered works of authorship,\textsuperscript{88} and recent cases confirm the copyrightability of computer programs.\textsuperscript{89} A computer program is considered a literary work of authorship\textsuperscript{90} to which copyright protection attaches.\textsuperscript{91}

\begin{footnotesize}
\begin{enumerate}
\itemsep-1pt
\item Cross-license demand. Judging from past events, most if not all significant patents in the software industry likely are to be cross-licensed eventually. Cook, supra note 69, at 53-54.
\item Id.
\item See Einschlag & Michaelson, supra note 5, at 406.
\item See Einschlag & Michaelson, supra note 5, at 406.
\item See Phillips, supra note 4, at 998; see also David R. Ellis, Computer Law—A Primer on the Law of Software Protection, 60 Fla. B.J. 81, April 1986 (stating that, "Perhaps the most important method of protecting computer software is through the law of copyrights .. .. ").
\item See Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983) (ruling that programs in both source code and object code were protected by copyright and rejecting the argument that object code should not be protected because it only communicates to a machine and is not readable by humans); Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982); Tandy Corp. v. Personal Micro Computers, Inc., 524 F. Supp. 171 (N.D. Cal. 1981).
\end{enumerate}
\end{footnotesize}
Section 102 of the 1976 Copyright Act defines the subject matter of copyright. Subsection (a) provides that copyright subsists "in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device." 92

A work of authorship must be "original" in order to qualify for copyright protection. The originality requirement has two facets: first, the author must have engaged in some intellectual endeavor and not just copied from a pre-existing source; second, the author must have expressed a minimal amount of creativity. Thus, the author must contribute more than just a trivial variation from a previous work. For example, copyright protection does not cover language that is cliched, that expresses an idea in a more or less stereotypical manner, or scenes a faire ("stock" scenes that are standard or necessary in treating a given topic). 93

The basic purpose of copyright law is to protect the expression of an idea, but not the underlying idea itself. 94 Copyright does not protect actual ideas, processes, procedures, methods of operation, systems, concepts, discoveries,

---


92 See Barrett, supra note 14, at 138.

According to the House Committee Report No. 1476, 94th Cong., 2d Sess. 54 (1976), reprinted in 1976 U.S. Code Cong. & Admin. News 5659, 5667 respecting the Copyright Act, literary works include 'computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas... Any doubts as to whether computer programs could be subject of copyright were dispelled by the passage of the Computer Copyright Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028 (1980). The effect of this brief amendment, according to the House Report, was to 'clearly [apply] the 1976 law to computer programs... H.R. Rep. No. 1307, 96th Cong., 2d Sess. 19 (1980), reprinted in 1980 U.S. Code Cong. & Admin. News 6492, 6509. The Computer Copyright Act defined computer programs as 'a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.


93 Id.

94 See Baker v. Selden, 101 U.S. 99, 102-106 (1879) (holding that the author of a copyrighted book describing a new bookkeeping method was not protected from others using the method; rather, the author was only protected from others reproducing his expression of the bookkeeping ideas).
principles, or utilitarian aspects of a work. While copyright law protects many different expressions of an invention, it does not protect the underlying functionality that usually constitutes the true innovation. Hence, copyright law extends protection to the software code itself from direct plagiarism, but it does not extend protection to the ideas behind the software. This allows competitors to legally duplicate underlying ideas so long as they do not imitate the expression of those ideas.

"The utilitarian aspect of object code was at one time suggested as a reason for not extending copyright protection to object code, but this reasoning was ultimately rejected." Proposed statutes would extend copyright-like protection to the functional aspects of software and algorithms. Congressional action could provide additional levels of protection to encourage innovation in these critical technological areas. In fact, the Supreme Court has consistently urged congressional action in this field. "Congress'
LACK OF PROTECTION AFFORDED SOFTWARE

failure to act, however, has left the functional aspects of software and algorithms unprotected. 102

Copyright law presently affords computer software the same protection as works of a creative nature such as books, plays, and musical recordings. 103 However, most consumers clearly "purchase software for its utilitarian value in accomplishing certain functions" as opposed to its artistic or expressive value. 104 Functionality and ease of use are far more important to a purchaser of software than its aesthetic appeal or originality. 105

Software producers spend considerable time, energy and resources developing marketable software products. They must rely on a steady stream of sales over the lifetime of the product in order to recoup their expenses and realize a profit. 106 Under the current copyright laws, however, competitors may copy the underlying idea of the software, and express it in their own fashion at a much lower end cost since they avoided the expensive developmental expenditures. Consumers are more likely to buy the cheaper copy, which would provide the same use, or perhaps, improved functionality over the original product. 107

"In contrast, many copyrightable works [(other than software)] are purchased for originality or aesthetic" value. 108 Consumers of such works are willing to pay premium rates for original artistic ideas. "[C]reative works have

---


The Copyright Act affords protection to "original works of authorship fixed in any tangible medium of expression." 17 U.S.C. Section 102(a).

This broad category of protected "works" includes "literary works," id., which are defined by the act as works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film tapes, disks, or cards, in which they are embodied. 17 U.S.C. Section 101. While computer programs are not specifically listed as part of the above statutory definition, the legislative history leaves no doubt that Congress intended them to be considered literary works. See H.R. Rep. No. 1476, 94th Cong., 2d Sess. 54, reprinted in 1976 U.S.C.C.A.N. 5659, 5667; Whelan, 797 F.2d at 1234; Apple Computer, 714 F.2d at 1247.

Altai, 23 U.S.P.Q. 2d at 1249.

104 Phillips, supra note 4, at 1009.

105 See id. at 1009. "Diversity is the primary goal when it comes to novels, songs, and other traditional domains of copyright. Readers want to read novels they have not read. But diversity is not the goal of interface design. Computer users want consistency in interfaces because this promotes ease of use." Richard Stallman & Simon Garfinkel, Against User Interface Copyright, COMMUNICATIONS OF THE ACM, Nov. 1990, at 16.

106 Phillips, supra note 4, at 1009.

107 Id. at 1010.

108 Id.
a natural disincentive for copying—if consumers feel that a particular expression is [overly] derivative of past works, they are not likely to be attracted to it." However, software that incorporates an excessive number of new commands and concepts may provide a disincentive to purchase because of the new learning curve that a consumer must undertake in order to master the new features. Thus, a new software product that possesses the same features and characteristics of an older product, already known to consumers, reduces the learning curve barrier. This feature along with a lower price or improvements makes the product very desirable to software consumers. "[T]he utilitarian nature of software requires a different type of protection than that afforded by the copyright law to creative-type expressions." Software architecture has been best described "as the overall structure and organization of the program" because the "architecture sets forth the operations, algorithms, routines and data structures, and establishes the logical relationships between them" that are necessary to effectuate the desired goal. The federal courts have yet to achieve consistent treatment of copyright protection for software architecture. In Whelan Associates, Inc. v. Jaslow Dental Lab., Inc., the Third Circuit's extension of copyright protection for computer software dramatically changes copyright law, "particularly with regard to the fundamental idea/expression dichotomy."

109 Id.
110 Id.
113 Whelan gave developers a substantial boost in their efforts to employ copyright law to protect program architecture. The Third Circuit held that one program infringed upon another solely because of similarities in structure. Id. Whelan Associates had written a program marketed as "Dentalab" for the IBM Series One computer. Id. at 1225. The program was billed as designed to meet the business needs of dental laboratories. Id. The program was developed through consultation with Jaslow Dental labs, which had helped identify these business needs. Id. Whelan Associates and Jaslow Labs entered into an agreement a year after the program became operational under which Jaslow Labs agreed to market the program. 797 F.2d at 1225. The president of Jaslow Labs developed a dental lab program in a language compatible with personal computers after discovering that many of the smaller dental labs employed personal computers on which Dentalab could not be used. Id. This software was marketed as "Dentcom PC." Id.

Whelan Associates immediately sued Jaslow Labs for infringing on their Dentalab copyright. Id. at 1227. At trial an expert witness testified that "substantive differences in programming style, in programming structure, in algorithms and data structure, all indicate that the Dentcom system is not directly derived from" Dentalab. Id. at 1228. The same expert witness' written report concluded, however, that the Dentalab and Dentcom PC programs shared "overall structural similarities." 797 F.2d at 1228. The district court relied on this finding in holding that the Dentcom system was substantially similar to Dentalab. Id. The district court held that Jaslow, misusing its access to the
Programmers seek to minimize the number of operations necessary to avoid exceeding memory restraints. Accomplishing the objective of achieving an optimal architecture for functions and data requires great ingenuity and considerable trial and error. This portion of the programming stage requires considerable time and investment so the true value of the program rests not in the actual code, but in the program's architecture.115

There has been widespread dissatisfaction with the Whelan court's formulation of the idea/expression dichotomy.116 The Whelan court's extension of copyright protection to software seriously distorts copyright law, especially with respect to the fundamental idea/expression dichotomy. Under Whelan, original program, had copied Dentcom PC from Dentalab, thereby infringing on Whelan Associates' copyright. Id.

114 See Kretschmer, supra note 111, at 823. One commentator has said, The idea/expression dichotomy is a doctrine that provides that only the expression of an idea may be subject to copyright, not the idea itself. Merger of idea and expression may be established by showing that there are only a limited number of ways in which the idea may be expressed. Where the "underlying ideas are capable of only a limited range of expression, they 'may be protected only against virtually identical copying.'" When the idea and its expression merge, becoming inseparable, copying the expression is not considered infringement since "protecting the 'expression' would confer a monopoly of the 'idea.'" Where merger exists, a finding of substantial similarity will be precluded.


115 See Pamela Samuelson, CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form, 1984 Duke L.J. 663, 688. Samuelson states, "On complex assignments, a programmer may produce, on an average, only a few words of code a day." Id. Most of the program development time is used to find creative solutions that maximize the speed of the program and minimize the size of the program. See generally Kretschmer, supra note 111, at 826 n.6.

116 Kretschmer explains,

Four basic and often interrelated criticisms have emerged. They are: (1) the Whelan formulation of the idea is conceptually overbroad and thereby misses the significance of the idea/expression dichotomy; (2) the Whelan approach fails to recognize that different ideas may underlie the different levels of the program; (3) the Whelan approach ignores the influence of market factors in determining structure and potentially grants the first developer of a program a monopoly over software in the field; and (4) any decisions about copyright infringement of software architecture based on the idea/expression dichotomy will inevitably be arbitrary, and so will provide little guidance to software developers and other courts. These substantial criticisms suggest that the idea/expression distinction should be altogether abandoned in favor of the conceptually neater and practically simpler utilitarian/aesthetic distinction employed by the court in Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003, 1010 (N.D.Tex. 1978) (the district court refused to extend copyright protection to the structural elements of software).

Kretschmer, supra note 111, at 839.
trial courts will be required to analyze software infringement cases using copyright law; this will require a case by case analysis due to the technological complexity of software infringement cases.\textsuperscript{117} Idea and expression are not distinct cognitive entities. Idea and expression represent opposite ends of the cognitive continuum; in the middle of that continuum, idea and expression become indistinguishable.\textsuperscript{118}

The Second Circuit declined to follow \textit{Whelan} in \textit{Computer Associates International, Inc. v. Altai, Inc.}.\textsuperscript{119} Instead the court employed a three-step procedure, based on the abstractions test\textsuperscript{120} utilized by the district court, to

\begin{itemize}
\item \textsuperscript{117}Id. at 823. One commentator has noted, [A]dvances in computer and software technology are eclipsing the industry's capability to bring items to market in a commercially usable form. Because the technological lifetime of a typical software product is only two to three years, it is highly unlikely that the slower-evolving judge-made law would be able to keep pace with these advances. For example, recent advances in artificial intelligence technology will obfuscate the difference between a program and the data on which it operates. These advances, and others yet unforeseen, will blur the distinction between idea and expression further to a point in which it may no longer have any useful meaning. Already, the idea-expression dichotomy used to determine the protectable aspects of a copyrighted work may yield completely arbitrary results when applied to computer programs. Phillips, \textit{supra} note 4, at 1026 (citations omitted).
\item \textsuperscript{118}Kretscher, \textit{supra} note 111, at 839 n.71. In \textit{Nichols v. Universal Pictures Corp.}, 45 F.2d 119 (2d Cir. 1930), Judge Learned Hand presented an abstraction test as the inspiration for the idea/expression continuum. Judge Hand's crucial observation was: Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his 'ideas,' to which, apart from their expression, his property is never extended. \textit{Id.} at 121.
\item \textsuperscript{119}23 U.S.P.Q.2d 1241 (2d Cir. 1992).
\item \textsuperscript{120}Id. at 1252. The \textit{Altai} court explained,
\begin{quote}
As applied to computer programs, the abstractions test will comprise the first step in the examination for substantial similarity. Initially, in a manner that resembles reverse engineering on a theoretical plane, a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it. This process begins with the code and ends with an articulation of the program's ultimate function. Along the way, it is necessary essentially to retrace and map each of the designer's steps—in the opposite order in which they were taken during the program's creation.
\end{quote}
\begin{quote}
As an anatomical guide to this procedure, the following description is helpful: At the lowest level of abstraction, a computer program may be thought of in its entirety as a set of individual instructions organized into a hierarchy of modules. At a higher level of abstraction, the instructions in the lowest-level modules may be replaced conceptually
\end{quote}
\end{itemize}
determine whether the non-literal elements of the computer programs were substantially similar. The court equated computer program design efficiency with deriving the most concise logical proof or formulating the most succinct mathematical computation. The more efficient a set of modules are, the more closely they approximate the idea or process embodied in that particular aspect of the program’s structure. The abstraction test systematically eliminates from copyright protection the efficient aspects of a computer program. What remains after the analysis is what should be protectable.

The *Altai* court basically adopted a test which rewards inefficient programmers over efficient ones. Competent programmers developing efficient programs are not protected under *Altai*, while less competent programmers drafting inefficient programs are protected. Certainly Congress did not intend to reward inefficient computer programs with copyright protection over efficient programs. The results of *Altai* exemplify how inappropriate copyright registration is for computer programs. The court in *Altai* acknowledged, "[W]e think that copyright registration—with its indiscriminating availability—is not ideally suited to deal with the highly dynamic technology of computer science." Software architecture deserves protection; copyright, however, is not the proper method for providing that protection. Trade secret protection is a more amenable method of protecting software architecture.

by the functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace implementations of those modules in terms of lower-level modules and instructions, until finally one is left with nothing but the ultimate function of the program. ... A program has structure at every level of abstraction at which it is viewed. At low levels of abstraction, a program’s structure may be quite complex; at the highest level it is trivial. ... Once the program’s abstraction levels have been discovered, the substantial similarity inquiry moves from the conceptual to the concrete. Professor Nimmer suggests, and we endorse, a "successive filtering method" for separating protectable expression from non-protectable material. *See generally* 3 Nimmer § 13.03[F]. This process entails examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain and hence is non-protectable expression. ... The structure of any given program may reflect some, all, or none of these considerations. Each case requires its own specific investigation.

*Id.* at 1253 (citations omitted).

121*Id.*

122*Id.*
C. Trade Secrets

Trade secrets123 are a nonstatutory area of law that can be employed to offer protection against theft of computer programs.124 "A trade secret is a formula, pattern, device or information which is used in the operation of a business and provides that business with an advantage or an opportunity to obtain an advantage over those who do not know or use it."125 In order to remain a secret, the owner of the trade secret must take precautions so the trade secret does not become accessible to individuals other than those expressly selected by the owner to be privy to it.126

The primary problem with trade secret protection is that the only feasible method to maintain a marketed program as a secret is to impose a contractual obligation of confidentiality on the purchaser. However, if the purchaser divulges the trade secret to a third party, the vendor will have no cause of action against that third party. Moreover, vendors are reluctant to sue their customers, which gives rise to the anomaly inherent in trade secret protection—that a vendor cannot sue its customers or third parties.127

IV. INCOMPATIBILITY OF THE VARIOUS FORMS OF PROTECTION

The current intellectual property regime in the United States limits the owner of software to one type of protection at a time. Invoking one type of intellectual property protection often precludes a software owner from using concurrent methods of protection. Copyright does not entirely preempt state trade secret. Registration, however, can destroy the secrecy required for trade secret protection since registered works are deposited with the Copyright Office and the Library of Congress, and are available for public inspection.128 As one

---

123See 1939 Restatement of Torts § 757, comment b.

124See Edwin H. Taylor, Protection of Computer Software, at 181, 188 (PLI Patents, Copyrights, Trademarks, and Literary Property Course Handbook Series No. G4-3831, 1989), available in WESTLAW, 269 PLI/Pat 181. (Computer software including both object code and source code clearly qualifies as subject matter protectable by trade secret law, both under the traditional definition set forth in the Restatement and the definition in the Uniform Trade Secret Act. Obviously, a key requirement is that of secrecy and this is discussed later. Constant care must be taken to keep software secret); see, e.g., Belth v. Insurance Dep't, 406 N.Y.S.2d 649 (Sup. Ct. 1977). State laws on trade secrecy have consistently been upheld to apply to computer software.

125See Ellis, supra note 86, at 82.

126Id.


128Jensen stated, 
... Office regulation which allows an exemption from the Library of Congress deposit requirement for a computer program when the program is published only in machine readable form and when
commentator notes, "It is impossible to retain trade secret protection for property that is the subject of a patent. The secrecy requirements of trade secret are diametrically opposed to the disclosure requirements for patent protection." In addition, the uncertainty associated with acquiring patent protection for computer software can leave a developer of software without either a patent or a trade secret after undergoing a long and expensive patent application process in which some or all of the secrets that were disclosed in the patent application become part of the public record through the opinions published by the courts when the denial of the application is appealed.

Copyright does not protect the ideas, algorithms and logic underlying software programs. Patent protection is very expensive and often difficult to obtain. Trade secret protection requires the owner to monitor the use of the program by licensees to make sure they are not misusing or disclosing the secret, which is a near impossible task when software is mass-licensed to hundreds or thousands of users. Additionally, the uncertain effects of copyright law on trade secret protection and the lack of adequate means to protect secrecy in the registration process compel developers to either weaken their copyright protection by refusing to register at all or by registering only the object code under the "rule of doubt," or to further weaken their trade secret protection by depositing source code information with the Copyright Office. Ironically, the uncertainty of obtaining patent protection and the risk of disclosure though appeals (even where protection is ultimately denied) force developers to risk loss of trade secret protection in order to pursue the mere possibility of patent protection. In summary, none of the methods on their own provide sufficient protection, and the protection schemes are incompatible for providing concurrent protection.

Identifying materials are submitted to the Copyright Office in lieu of the entire source code. The identifying materials which must be submitted consist of the first and last twenty-five pages of the source code and the page containing the copyright notices. The code can sometimes be arranged so that the first and last twenty-five pages contain only nonsecret information which will not destroy the secrecy of the trade secret, but in the case of short programs, even the identifying materials requirement may require the deposit of all or substantially all of the source code. In summary, the only sure registration of a computer program is the submission of at least the first and last twenty-five pages of the source code, which is likely to disclose enough of the program to destroy the secrecy required for trade secret protection.

Jensen, supra note 42, at 1446.

129 Id. at 1446.

130 Id. When a patent application is denied, the applicant is caught in a Catch-22 situation—he must either forgo his/her right to an appeal and give up all hope of ever obtaining a patent, or take his appeal and hope that his secrets will not be disclosed in an opinion or that he will be able to seal the record before the opinion is released.

131 Id. at 1433.
V. Conclusion

A new form of intellectual property protection "sui generis" is needed to balance the interests of software developers with those of society.132 The Semiconductor Chip Protection Act of 1984133 was enacted to provide sui generis protection for micro chip mask works in order to curtail rampant pirating. Such protection is necessitated by the extraordinary amounts of self-help that developers must pursue because of the lack of protection and policing mechanisms under the current intellectual property regime.134

Our nation's economic viability is dependent on advances in technology. Advancement in the computer software industry is seriously hampered by the inability of our intellectual property laws to adequately protect software innovations. Congress has left the software industry in a quagmire by manipulating the traditional forms of intellectual property protection to cover software. New technologies do not fit neatly into the existing intellectual property framework.135 Due to the importance of the computer software industry, it is reasonable to provide independent treatment to such technology.

Endless issues must be addressed by Congress in providing adequate protection for computer software. Some items that should be considered are: (1) submitting, during filing, a complete copy of the source code to be protected; (2) shortening the scope of protection to one or two years; (3) increasing the number of examiners in the software area in order to reduce prosecution time; (4) imposing stricter penalties for computer software piracy and infringement; and, (5) granting monopolies on complex algorithms, such as Professor Bracewell's Fast Fourier Transform, with exemptions for human thought processes and academic research.136

A new method of protection must be instituted in order to provide incentive to software developers to invest time, money and resources on software


134See James A. Eidelman & Carol R. Shepherd, Living Among Pirates: Practical Strategies to Protect Computer Software, MICH. B.J. (Mar. 1986). Many developers are employing means such as: (1) physical copy protection-to prevent a program disk from being copied using the computer's normal copy procedures; (2) embedding the end user's name in the program so that it will be displayed on the screen, reminding him/her of the personal nature of the license; (3) embedding each distributed program copy with its own serial number. By requiring the return of a signed agreement with the serial number as a condition to providing valuable product assistance and support, the client discourages theft and may trace illegal copies to users responsible for piracy; (4) shrink wrap license agreements.

135See Phillips, supra note 4, at 1036-37.

136See id. at 1038-42 (listing more items that should be addressed in sui generis legislation).
innovation. We are entering a global economy; our country cannot afford to handicap its software industry by employing archaic laws that are ill-equipped to handle such high technology as computer software.

HIMANSHU S. AMIN
1995 JUDGE JOHN M. MANOS
WRITING COMPETITION ON EVIDENCE

First Place

Kathy Jo Cook
(Suffolk University Law School)

Second Place

Fred J. Hjelmeset
(University of California - Hastings College of Law)

Honorable Mention

William L. Bernard
(Hamline University School of Law)

Robert F. Croskery
(University of Cincinnati College of Law)

Varsha Kukafka
(Northeastern University School of Law)

Stacy M. Lemming
(Case Western Reserve University School of Law)

Paul T. Olszowka
(University of Wisconsin Law School)

Richard Selinger
(Western New England School of Law)

The Cleveland State Law Review would like to thank the judges of the 1995 writing competition:

Lloyd B. Snyder, Associate Professor of Law, Cleveland-Marshall College of Law

Alexander Rokakis, Assistant United States Attorney, Northern District of Ohio

Hollie L. Gallagher, Cleveland State Law Review

Claudia Sanjour, Cleveland State Law Review

Eric F. Spade, Cleveland State Law Review

Stacy Zingler, Cleveland State Law Review