Technological Poetry: The Interface between Copyrights and Patents for Software

Willis E. Higgins

Follow this and additional works at: https://repository.uchastings.edu/hastings_comm_ent_law_journal

Part of the Communications Law Commons, Entertainment, Arts, and Sports Law Commons, and the Intellectual Property Law Commons

Recommended Citation
Available at: https://repository.uchastings.edu/hastings_comm_ent_law_journal/vol12/iss1/3

This Article is brought to you for free and open access by the Law Journals at UC Hastings Scholarship Repository. It has been accepted for inclusion in Hastings Communications and Entertainment Law Journal by an authorized editor of UC Hastings Scholarship Repository. For more information, please contact wangangela@uchastings.edu.
Technological Poetry: The Interface Between Copyrights and Patents for Software

by
WILLIS E. HIGGINS*

Introduction

Both the U.S. copyright laws¹ and the U.S. patent laws² are constitutionally mandated as follows: "The Congress shall have Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."³ By the conventional grammatical rule of parallelism, "Science" is related to "Authors" and "Writings," and "useful Arts" is related to "Inventors" and "Discoveries." These categories work well for most copyrightable and patentable subject matter because literary works and technological works are generally exclusive of each other.⁴

Computer software, however, presents an interesting overlap of the two categories because it has both literary and technological components. Well-written software shares with poetry the characteristic of conveying significant meaning with a minimum number of words, hence the characterization "tight code."⁵ The literary component of software⁶ also refers to the manner in which it is created (i.e., it is written by the programmer). However, unlike other literary works, software also has a technological component, which causes a machine, computer hardware, to operate for the purpose of executing a myriad of tasks. The dual nature

---

* Flehr, Hohbach, Test, Albritton & Herbert, San Francisco, California. The assistance of Steven J. Adamson, Gary S. Williams and Stephen C. Durant in the preparation of this article is acknowledged.
5. "Tight code" is concise, well-organized, efficiently-running code.
of software allows software developers to obtain both copyright and patent protection. This Article discusses the nature of the protection that is, and should be, afforded by both copyright and patent laws. The Article explains the advantages of patent protection for software and offers practical suggestions for preparing software applications.

I

Election of Protection

Although there are no cases so holding, some authorities have argued that copyright and patent protection for software should not coexist and that a software author/inventor should be required to elect one form of protection. The argument is grounded on the proposition that, if the software is also protected by copyright, then there is a lack of consideration for the patent "contract", granting exclusive rights for a limited period in return for full disclosure of the invention. Furthermore, unless the coverage of the copyright is identical to that of the patent, it cannot be said that there is no consideration for the patent grant. Thus, the argument appears to assume that a patent and a copyright on the same software are coextensive in scope.

This position ignores the dual nature of computer software. There is no reason why the same program cannot or should not potentially have its literary component protected by copyright and its technological component protected by patent. As will be seen, patent protection is almost always broader in scope, and is always different in scope. Thus, it would be possible to practice an invention formerly covered by an expired patent on a program without infringing a copyright still in force on the same program.

A. Nature of Copyrights

The procedural characteristics of the copyright system have important implications for its role in computer software protection. The copyright registration process contains no mechanism for defining the scope of the copyright. The registration application is examined only to determine if the material deposited constitutes copyrightable subject matter and if other legal and formal requirements of the copyright laws have
been met. Although a copyrightable work must satisfy a requirement of originality, no search of prior works is performed by the Copyright Office. In applications for protection of a compilation or derivative work, the applicant must only identify the pre-existing work and generally describe the added material. In the case of software, a deposit of the complete work for which copyright is claimed in the registration application is often not required; a deposit of "identifying portions" is sufficient.

Although the copyright statutes declare that protection is limited to expressions and does not extend to underlying ideas, procedures, processes, systems, methods of operation, concepts, principles or discoveries, the registration process does not implement these rules. Copyright registration has been characterized as "largely a formality," since most copyrights are registered within two or three months of filing the application.

In summary, copyright registrations are easily and quickly obtained, but uncertain in scope. Copyright registration is also inexpensive. A registration application fee of $10, an application on a form prescribed by the Register of Copyrights, and one or more copies of the work to be copyrighted (unless exempted by the Register of Copyrights) are all that is required. Although registration is not necessary for copyright protection, it is required in most instances prior to filing suit, and provides advantages in the remedies that can be obtained for infringement.

B. Nature of Patents

The patent system procedural and substantive requirements are quite different in nature. The subject matter which potentially qualifies for patent protection is "any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof . . . ." An applicant must file a patent application with the U.S.

15. A.B.A. 1989 Committee Reports, supra note 7, at 388.
Patent and Trademark Office that makes a complete disclosure of the invention, sufficient for someone skilled in the art to practice the invention.\textsuperscript{23} The best way the applicant knows to practice the invention must be disclosed at the time of filing.\textsuperscript{24}

The application must include one or more claims particularly pointing out and distinctly claiming what the applicant regards as the invention.\textsuperscript{25} These are one-sentence statements listing each element of the invention. In the case of software, the elements are either parts of a machine (the data processing system on which the software runs) or process steps defining the sequence of operations that takes place when the software runs. The claims must describe the invention in a way that defines a patentable difference over previous knowledge (the prior art). To assist in the examination process of the application, the applicant must disclose any prior art known to the applicant, which is material to the examination process.\textsuperscript{26} In addition to being different than the prior art,\textsuperscript{27} the defined invention as a whole must not be obvious to a person of ordinary skill in the art.\textsuperscript{28} A filing fee of at least \$300 is required.\textsuperscript{29} The preparation of the application, particularly drafting claims that protect the invention properly and meeting Patent and Trademark Office requirements, is an exacting and specialized task.

The application is assigned to a patent examiner, based on its claimed subject matter. The examiner carries out a search of the prior art and prepares an office action, rejecting at least some of the claims as anticipated or obvious over the prior art, and sometimes rejecting the claims for failure to define the invention with sufficient clarity.\textsuperscript{30} The applicant then responds, usually changing the wording of the claims to define the invention more clearly or to further distinguish it from the prior art. The applicant should stress that the invention as claimed is not taught or suggested by the prior art, and otherwise meets the requirements for a patent.\textsuperscript{31} When the examiner finds that the application meets the standards for granting patent approval, a written notice of allowance is given or sent to the applicant specifying the remaining fees necessary to issue the patent.\textsuperscript{32}

\textsuperscript{23} 35 U.S.C. § 112.
\textsuperscript{24} 35 U.S.C. § 112.
\textsuperscript{25} 35 U.S.C. § 112.
\textsuperscript{26} 37 C.F.R. § 1.56 (1988).
\textsuperscript{27} 35 U.S.C. § 102.
\textsuperscript{29} 35 U.S.C. § 41(a).
\textsuperscript{30} 37 C.F.R. § 1.101–.106.
\textsuperscript{31} 37 C.F.R. § 1.111.
\textsuperscript{32} 37 C.F.R. § 1.311(a); 35 U.S.C. § 151.
Preparing and prosecuting a patent application for software typically costs up to $10,000\textsuperscript{33} and may be more for a complex program. Completing the application process usually takes two to three years.\textsuperscript{34} In summary, obtaining a patent is an expensive and fairly rigorous procedure to ensure that the invention is both novel and nonobvious. Once granted, a patent gives the patent owner "the right to exclude others from making, using or selling the invention throughout the United States" for seventeen years.\textsuperscript{35}

II

Historical Background

In principle, there should be no overlap between the literary coverage of copyrights and the technological coverage of patents. The literary/expressive aspect of a software program should be protected by copyright, and the innovative/technological aspect by patent. The two areas of coverage are distinct, as are the procedures for obtaining protection. There was a substantial period of time, however, during which the patentability of computer software was highly doubtful.\textsuperscript{36}

Instead of granting patent protection for computer software, the courts expanded the scope of copyright protection, thus creating an overlap. Thus, there is an overlap for historical reasons. The history of patent protection for software has been reviewed elsewhere.\textsuperscript{37} Briefly, the Patent and Trademark Office opposed patenting software because of bureaucratic concerns over its ability to handle software patent applications.\textsuperscript{38} Perhaps persuaded by these concerns, the Supreme Court sharply limited the availability of patent protection for software in a series of decisions during the 1970s.\textsuperscript{39} These decisions narrowly interpreted "machine" and "process" in the statutory subject matter of

\textsuperscript{33} Haynes and Durant, Patents and Copyrights in Computer Software Based Technology: Why Bother With Patents?, 4 COMPUTER LAW. 1, 3 (1987).

\textsuperscript{34} COMMISSIONER OF PATENTS AND TRADEMARK ANN. REP. 21 (1987).


\textsuperscript{36} Much software was considered not to come within the statutory classes of subject matter that could be patented. See H.R. REP. No. 781, 98th Cong., 2d Sess. 8-11 (1984).


\textsuperscript{39} See, e.g., Gottschalk, 409 U.S. 63 (denying patent protection to laws of nature and mathematical expressions because allowing protection would be impractical and would suppress technological progress); Dann v. Johnston, 425 U.S. 219 (1976) (process was obvious computer technology, if not obvious to the banking industry); Parker v. Flook, 437 U.S. 584 (1978) (denying patent because the only novel feature was a mathematical formula, but cautioning that the decision did not hold that patent protection "of certain novel and useful com-
Inventions. In 1972, the Supreme Court in *Gottschalk v. Benson* held that a patent could not preempt an algorithm, mathematical formula, or calculation. The Court defined an algorithm as a procedure for solving a given type of mathematical formula. Many thought the Court's decision and reasoning contrived. In later cases, the Court, following *Benson*, denied patents in *Dann v. Johnston* and *Parker v. Flook*. These decisions displayed the Court's confusion of the statutory requirements of section 101 of the Patent Act with those of section 102 (novelty) and section 103 (nonobviousness) by including in the subject matter test of section 101 the requirements of novelty and nonobviousness. Finally, in 1981 the Supreme Court in *Diamond v. Diehr* awarded a patent for the first time to a computer-related invention that included an algorithm. While patents were granted prior to this case, they were limited to programs without algorithms. The *Diehr* decision cleared up the confusion of prior court decisions and dictated a new method of examination. The Court held that the applicant's software should be examined as a whole, instead of individually examining the mathematical equations. In later cases, courts further clarified the meaning of algorithm. Today, following these cases, software claims are eligible for patent protection unless they involve the use of a simple mathematical formula to calculate and display a number.

In *Diehr*, the Court adopted a broad view of the subject matter that qualifies for patent protection. While the case involved a computer-controlled rubber curing process, the holding in *Diehr* marked the first time that a patent claim was held to be proper even though it was a computer-related invention that included an algorithm. Following *Diehr*, the Court of Customs and Patent Appeals, the predecessor of the present computer programs will not promote the progress of science and the useful arts or that such protection is undesirable as a matter of policy.

41. 409 U.S. 63 (1972).
42. Id. at 65.
43. See generally Maier, supra note 37.
45. 437 U.S. 584 (1978).
47. The Court found mathematical equations were the building blocks of invention and remain unpatentable in their basic form. The *Diehr* decision also held these equations placed in combination with one another to solve a mathematical formula (an algorithm) were patentable for a specifically claimed method of use, but not for a general one. 450 U.S. 175 (1981).
49. *In re Pardo* 684 F.2d 902 (C.C.P.A. 1982); *In re Abele* 684 F.2d 912 (C.C.P.A. 1982).
Court of Appeals for the Federal Circuit, decided several cases in which it developed a two-pronged test that narrowed the prohibition against algorithms. As a result of these decisions, most software is now proper subject matter for patent protection. To be patentable, of course, software must also satisfy the novelty and nonobviousness requirements. The remnant of the former law against patenting software denies patent claims directed solely to using a mathematical formula or algorithm to calculate and display a number. Even this restriction, however, may have little practical effect in view of the recent increase in software patents being issued by the Patent and Trademark Office.

Two very recent cases demonstrate the fine distinctions between statutory and nonstatutory claims involving mathematical algorithms. In re Grams held claims reciting a data analysis mathematical algorithm nonstatutory on the ground that the sole process step other than the algorithm merely recited data collection for the algorithm. In re Iwahashi held claims reciting an autocorrelation mathematical algorithm to be statutory subject matter for patent protection because the claim recited several apparatus elements in addition to the mathematical algorithm. The result would apparently have been the same if the claim had been to a process reciting several process steps in addition to the mathematical algorithm, such as steps requiring that operations be carried out on physical objects, including conventional computer hardware. The lesson in these two cases is that patent claims reciting mathematical algorithms must be carefully drafted to be considered statutory.

The Copyright Office first took the position that software could be copyrighted as early as 1964. The legislative history of the 1976 revision of the copyright law shows that Congress intended computer pro-

50. See, e.g., In re Freeman, 573 F.2d 1237 (C.C.P.A. 1978); In re Walter, 618 F.2d 758 (C.C.P.A. 1980); In re Pardo, 684 F.2d 912 (C.C.P.A. 1982); In re Abele, 684 F.2d 902 (C.C.P.A. 1982).
51. See, e.g., MANUAL OF PATENT EXAMINING PROCEDURE § 2106 (Rev. 12) (July 1989) (setting forth guidelines for the examination of software related patent applications). See also Notice Patentable Subject Matter-Mathematical Algorithms and Computer Programs, 1106 TMOG 5 (September 5, 1989) (extending the scope of the above guidelines).
grams to be covered as literary works. Congress created the National Commission on New Technological Uses of Copyrighted Works (CONTU), which recommended that the copyright law be amended to explicitly state that computer programs are copyrightable subject matter. Congress responded by amending the Copyright Act of 1976 to conform to the CONTU recommendations.

In practice, from 1964 until well after the Diehr decision in 1981, most of the software community viewed copyright as the only form of intellectual property protection available for software. Given this situation, it is not surprising that courts broadened the scope of software copyright protection in order to protect against misappropriation. In Whelan Associates v. Jaslow Dental Laboratories, the court held that anything more specific than the general purpose of a software program is protected expression. In order to do this, the court defined the idea in very broad terms: "[t]he purpose or function of a derivative work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of that idea." Whelan has been the subject of much critical comment, and it has been both followed and rejected in subsequent decisions.

Most commentators agree that the Whelan test is overbroad, but no one has been able to articulate a meaningful test for determining when infringement of a software copyright occurs. Because the copyright system lacks a suitable procedure for defining the protected scope of

64. 797 F.2d 1222 (3rd Cir. 1986), cert. denied, 479 U.S. 1031 (1987).
65. 797 F.2d at 1236 (emphasis added).
68. See, e.g., Plains Cotton Coop. Ass'n v. Goodpasture Computer Service, Inc., 807 F.2d 1256 (5th Cir. 1987) (finding the similarities between the programs to be imposed by the externalities of the cotton market).
69. See, e.g., Petraske, supra note 66; Plains Cotton Coop., 807 F.2d at 1256.
technical subject matter, it seems likely that the legal system will be unable to articulate a meaningful standard for determining infringement when the infringement issues involve functional aspects of software. It is ironic that the copyright system has been thrust into this role by the workload concerns of the Patent and Trademark Office. In effect, a judge or jury in a software copyright infringement action in federal court must exercise the role of a patent Examiner and determine de novo the scope of a software copyright.

Intellectual property counsel find it very difficult to give their clients definitive opinions on copyright infringement, given the current state of the law. As long as the courts continue to provide protection for the functional aspects of software through copyrights, counsel must also assume the role of an examiner to determine the scope and content of the prior art, compare copyrighted software against that prior art to define a scope of protected expression, then compare the possibly infringing software to see if it comes within the defined scope. Attorneys trying software copyright infringement cases must prepare and try them in the same manner as patent infringement cases, but with the added burden of having to define the scope of protection for the court by, in effect, writing claims defining the scope. The recent cases expanding the scope of copyright protection to cover functional aspects of software can be viewed as filling the vacuum left by the perceived inapplicability of patents to software. While the tests the courts set forth may appear to be overbroad, they may be ultimately limited to copyright issues during the period that patent protection was believed not to be available.

In time, knowing that patent protection is potentially available, courts may tire of exercising the role of examiners to define the scope of copyright protection and say that the patent system must be used if broad protection on functional concepts of software is desired. Some commentators believe this is already beginning to happen.

The change in interpretation of the patent statutes by the courts has now filled the gap in protection for the functional aspects of new and nonobvious software. Intellectual property counsel are able to review the scope of protection defined by the claims in issued software patents. This defined scope, the administrative record and prior art considered in the Patent and Trademark Office, and additional prior art not considered in the Patent and Trademark Office, facilitates determining infringement.

73. These are called the "file wrapper" in patent jargon.
They allow intellectual property counsel to give definitive opinions on infringement questions and ease the burden for courts to decide infringement questions. The absence of such an administrative record makes determining the scope of a software copyright much more difficult. The description of the protected expression serves the function of a patent claim to define the scope of protection. The recent decision in *Manufacturers Technologies, Inc. v. CAMS, Inc.* illustrates, in its exhaustive analysis of the similarities between the copyrighted software and the software accused of infringing the copyright, how difficult the process of describing the protected expression and using the description to decide infringement can be when it is done carefully.

The patent system protects any technological innovation representing a sufficient departure from the prior art to warrant the grant of the right to exclude others from making, using, or selling it. The copyright system protects literary expression against unauthorized copying. Keeping these basic concepts in focus will allow each system to provide an appropriate level of protection with more efficient use of legal resources. While patent protection for interfaces is increasingly available, the Copyright Office is showing a reluctance to register copyrights on user interfaces in the form of menu screens.

## III

**Sui Generis Protection**

Some authorities have suggested that software requires a new form of intellectual property protection to balance the needs of the creator for protection and the interests of society to make use of knowledge. Such new forms of intellectual property protection are called sui generis protection. For example, one author argues on the basis of an economic analysis of software that a form of petty patent, having an expedited examination and a shorter term than a conventional utility patent, should be granted on operating system software. Furthermore, a modified form of copyright with a shortened term should be granted on application programs. An analysis of issued software patents, however, shows

---

77. See Durant, Patent Protection for Computer User Interfaces, 6 COMPUTER LAW., May 1989, at 12. (discussing the applicability of these basic concepts to computer user interfaces)
80. Id. at 1371.
81. Id.
that a high enough level of technical innovation can be found in both
categories for patent protection.  

Although particular programs may be short-lived, important technical
concepts used in successor programs may have a much longer useful
lifetime. By requiring that a patentable invention advance the art and by
providing a fairly difficult application procedure, the patent system at-
ttempts to limit its protection to innovations having a fairly high level of
technical merit compared to the prior art. This approach has served well
for hardware. The burden should be on those who assert that software
should be treated differently to establish that software differs from other
technical subject matter and warrants different treatment. Absent a
consensus from the public, the software industry, and the legal system
concerning the form that new protection should take, it is questionable
whether a new form of protection could be established to provide a better
level of certainty and superior results than can be achieved through com-
mon law evolution of the patent and copyright systems.

Such proposals for new forms of intellectual property protection for
software are prompted by the Semiconductor Chip Protection Act. However, this Act providing sui generis protection for mask works was
enacted five years ago, and it is still not clear whether it provides an
improved system of protection for that class of technological works. No
cases have yet been decided under it. Thus, a new system exclusively for
software protection may not provide better protection than the current
patent and copyright laws.

IV
Strategy and Tactics for Patenting Software

Two empirical studies, one of software patents issued over a ten-year
period and the other of patents issued over a six-month period, provide an indication of the kind of software being patented and by whom.
The ten-year study found that the software patents issued from 1978 to
1987 comprised 182 operating system software patents and 78 applica-
tions software patents. The operating system software included opera-
tions control and monitoring, data/file management, compilers, program
protection, security, and communications linking. The applications

83. See generally Samuelson, supra note 71, at 78.
§§ 901-14 (Supp. 1987)).
85. Soma & Smith, supra note 82.
86. Syrowik, Patent Protection for Software Technology—A Powerful New Form of Protec-
software included process control and monitoring, product design, graphics or imaging, word processing, sorting, medical diagnosis, business data analysis, geophysical data analysis, machine simulation, chemical analysis, and navigation calculations. The six-month study found 43 "pure" software patents issuing between July 1 and December 31, 1987, with slightly under half of the patents being applications software. These studies show that a significant number of software patents on a wide variety of software are now being issued by the Patent and Trademark Office.

Specific examples of issued patents on software include a menu system for a word processing system, windowing techniques, file systems useful in hypertext applications, identification of differences in words and sentences in documents (useful for red-lining programs), a resource allocation algorithm, and a Fast Fourier Transform algorithm, known as the discrete Bracewell transformation. The number and variety of patents now being issued on software mean that essentially any program that includes a new and nonobvious feature, which would likely be useful for longer than a short time, should be considered as a candidate for patent protection. The fact that the particular program may be short-lived is not determinative; the feature may be carried through to later versions of the program or to different programs.

A patent application must include both an enabling disclosure and a disclosure of the best way known by the inventor for implementing the invention. These are different requirements. A description can be enabling without disclosing the best mode known to the inventor for practicing the invention. These requirements can be satisfied with the use of block diagrams, flow charts, and psuedo-code. In many cases, this kind of documentation may need to be especially prepared for the patent application. An alternative is to disclose source code. Heavily commented code is particularly helpful. Disclosing source code may significantly reduce the cost of having the patent application prepared, but it does com-

87. Soma & Smith supra note 82, at 424-25.
88. Syrowik, supra note 86, at 972.
promise the trade secret status of the source code. For this reason, many applicants are reluctant to include source code in their applications.

The source code may be disclosed in the patent application and maintained as a trade secret until the patent is issued, which typically takes two to three years. Once the patent issues, the trade secret status of the source code in the application is lost. In many cases, the time that the patent application is pending will be all the time required for protection. The applicant, however, can balance the value of the coverage allowed by the Patent and Trademark Office at the conclusion of prosecution of the application against the value of maintaining the trade secret. If the applicant does not pay the issue fee for the application, it will be abandoned without disclosure of the source code.

If the applicant decides not to disclose source code, the description of the invention should be prepared very carefully. It is better to err on the side of including too much description than not including enough. If a patent application is filed just before a program is released, more than one year may elapse before a description is held to be inadequate by the Patent and Trademark Office. After the software has been commercially available for more than a year, it is too late to refile the application for the purpose of providing a more complete description. If source code is not disclosed, the issued patent may be subject to attack in litigation for failure to disclose the best mode for practicing the invention. A clear and complete technical description is therefore especially important in the absence of source code.

A patent application includes a set of claims after the technical description. The claims define the scope of protection sought in the application. For software, the claims are written as a system or apparatus, usually consisting of several elements in the form of means for carrying out certain functions, or as a process consisting of a sequence of steps carried out by running the software. It is often possible to obtain broader coverage with process claims because they do not include structural limitations present in system or apparatus claims. Some recent court decisions have interpreted system or apparatus claims of the kind used to define software inventions narrowly. These narrowing interpretations have not been applied to process claims. It is therefore important to include process claims in software patent applications.

Software patents usually contain a substantial number of claims defining the invention in different ways and with different scope of protection. Some of the claims are independent and some are dependent, i.e.,

referring back to one or more previous claims. Both independent and dependent claims are included because of uncertainty regarding the scope of claims allowed by the Patent and Trademark Office and to provide coverage of varying scope for the invention. More significant prior art may be discovered after an application has been issued as a patent. This prior art may invalidate the broader claims of the patent. The narrower claims may not be invalidated by such art and may still protect the commercially important forms of the invention.

Providing claims of different scope is especially important in software patent applications. The claims allowed by the Patent and Trademark Office will depend on the prior art available to the examiner in the prosecution of the application. Much software prior art is available only in commercial software; it has not been patented or published in the technical literature. If neither the applicant nor the examiner is aware of such prior art, it will not be considered, and some of the claims allowed will be too broad. Without claims of varying scope in the issued patent, such prior art could invalidate all of the claims in the patent, even if there are significant differences in the way a concept has been implemented in the patent and in the prior art found after the patent has issued. 98

V

Conclusion

Copyright protection should always be obtained for software because it is low in cost and provides effective protection against literal copying or other misappropriation of literary aspects of the software. While some courts have extended copyright protection to cover functional aspects of software as well, it is dangerous to rely solely on copyright to protect the manner in which a program will work. Such technological aspects of software are now clearly protectable with patents, which provide a defined scope of protection in their claims. With the availability of patent protection for software, courts may become increasingly reluctant to use copyrights to protect functional features of software.

98. The above discussion has been based on U.S. law. While the law regarding the patent protection of software in other countries has not developed as completely as in the United States, the trend in at least the developed countries is to follow the U.S. law. See generally Sumner & Plunkett, Powerful New Software Protection in Europe: The Patent Trend Continues, 4 COMPUTER LAW., October 1987, at 1. Decisions on international protection should therefore be based on current U.S. law. Even if the law of a particular country does not appear to allow patent protection for software at the time of filing, that law may change by the time the application is examined.