1-1-1988

Reflections on the State of American Software Copyright Law and the Perils of Teaching It

Pamela Samuelson
Berkeley Law

Follow this and additional works at: https://scholarship.law.berkeley.edu/facpubs
Part of the Law Commons

Recommended Citation

This Article is brought to you for free and open access by Berkeley Law Scholarship Repository. It has been accepted for inclusion in Faculty Scholarship by an authorized administrator of Berkeley Law Scholarship Repository. For more information, please contact jcera@law.berkeley.edu.
Reflections on the State of American Software Copyright Law and the Perils of Teaching It

by PAMELA SAMUELSON*

Teaching software copyright law in the United States is at present a perilous endeavor. Only one who does not teach the subject can think this assertion is an exaggeration. This essay will discuss the reasons for this peril.

First, almost all of the important questions about what copyright protection means for software have yet to be answered definitively. The copyrightability of computer programs is the most settled issue in this body of law1—but even it may not be completely settled if the NEC/Intel decision upholding the copyrightability of microcode is appealed.2 Although the copyrightability of a hardware version of a computer program has yet to be litigated, there is some chance that this issue will find its way into the courts.3 After all, if software can as easily be implemented in hardware as in electronic form, and if a copyright can issue for the software, why can't it issue for the equivalent hardware? There is an even greater likelihood that the

---

* Professor of Law, University of Pittsburgh School of Law. This article is a revised version of a paper delivered at the Conference of the International Association for the Advancement of Teaching and Research in Intellectual Property, which met at George Washington University School of Law on July 24-27, 1988.

1. Challenges to the copyrightability of computer programs were made in several cases in the early 1980's. See, e.g., Tandy Corp. v. Personal Micro Computers, Inc., 524 F. Supp. 171 (N.D. Cal. 1981); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), cert. dismissed, 464 U.S. 1033 (1984); Apple Computer, Inc. v. Formula International, Inc., 725 F.2d 521 (9th Cir. 1984). After both the Ninth and Third Circuit Court of Appeals' decisions upholding the copyrightability of computer programs, the copyrightability issue was, at least for applications and operating systems programs, no longer raised as a defense in software copyright infringement cases. But see Samuelson, CONTU Revisited: The Case Against Copyright Protection For Computer Programs in Machine-Readable Form, 1984 DUKE L.J. 663 (1984).

2. NEC Corp. v. Intel Corp., No. C-84-20799-WPG (N.D. Cal. Feb. 7, 1989) (ruling that microcode is copyrightable, but that NEC did not infringe Intel's copyright). A prior decision in the same case concerning the copyrightability of microcode can be found at 645 F. Supp. 590 (N.D. Cal. 1986), vacated in 835 F.2d 1546 (9th Cir. 1988) (The trial judge was excused from the case because of ownership of Intel stock and, on this ground, his prior order was vacated).

courts will someday have to deal with a copyright infringement claim based on a hardware "version" of a copyrighted piece of computer software.4

Even assuming that copyrightability issues have been put to rest, the more important issue is what, besides the exact sequence of bits constituting the program, a copyright does or should protect. It is fair to say that there is no consensus yet on this subject and that views about the proper scope of copyright protection for software vary about as widely as they possibly could.

Some would limit the reach of the copyright to the literal code. This "minimalist" position rests on the theory that because software is a technology, traditional copyright doctrine would require that the layer of protection be thin. The public policy goal of protection for software copyright under this theory is a very limited one: to prevent outright piracy of the code.5

In stark contrast to the minimalist view, some would make everything about a program, except its general purpose or function, an expression protected by copyright law. The "maximalist" view criticizes the minimalist position by asserting that if copyright law does not protect more than the literal code—if it does not also protect the more abstract creative parts of the design of software products—investment in software development will be discouraged. The maximalist view also attempts to ground itself in traditional copyright doctrine, relying heavily on "literary work" and compilation case law that endorses the protection of abstract structural features of copyrighted works.

The Third Circuit, in Whelan Associates v. Jaslow Dental Laboratories, has given the strongest expression of the maximalist position.6

4. The interchangeability of hardware and software has long been noted. See, e.g., Sprowl, Proprietary Rights in Programmed Computers: Looking Beyond the Hardware/Software Distinction for More Meaningful Ways of Characterizing Proprietary Interests in Digital Logic Systems, 1983 Ariz. St. L.J. 785, 785-86 (1983). A somewhat similar issue, raised in the patent context, was decided recently by the Court of Appeals for the Federal Circuit. The issue there was whether a software version of patented hardware was an infringement of the patent. See Pennwalt Corp. v. Durand-Wayland, Inc., 833 F.2d 931 (D.C. Cir. 1987), cert. denied, 108 S. Ct. 1226 (1988), cert. denied, 108 S. Ct. 1474 (1988). The court decided that the software was not equivalent to the hardware. It was an en banc decision, and the court was deeply divided. This decision may signal that claim drafters will need to be more careful in subsequent patent applications.


6. Whelan, 797 F.2d 1222. See also, Clapes, Lynch & Steinberg, Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs, 34 UCLA L.
The court's sweeping pronouncements in *Whelan* about the extent of copyright protection for software went far beyond the specific issues presented by the facts of that case, which largely focused on the protectability of the "structure, sequence, and organization" of software. In its holding, the court not only took the radical step of including the structures of software within the scope of copyright, but it also attempted to make a definitive statement on all software issues. The court announced a universal "test" for software copyright infringement, which would make everything about a program except its general purpose a protectable "expression." This general rule was to be subject to an exception if there were only a few ways to achieve the function.

Although most commentators seem to think that it is in the middle ground between these extremes of the "code-only" or "everything-but-the-general-purpose" positions that the right balance is to be struck in software cases, no one has been able to articulate how or where to find the elusive line between "idea" and "expression" for software: a line that will allow the right balance to be struck. It is, as always, easier to criticize the analysis or outcome of certain cases, or theories of other commentators, than to articulate a comprehensive analytic framework within which to make software copyright infringement determinations.

Overbroad decisions, such as that of the Third Circuit in the *Whelan* case, have set off new rounds of litigation by software industry leaders intent on forcing their competitors to compete by being more different. Although the Fifth Circuit, in the *Plains Cotton* case, explicitly rejected the *Whelan* test for software copyright infringement, as well as its more specific holding on the protectability of software

---

7. *Whelan*, 797 F.2d at 1224. Similarities between the two programs seem to have been found as to their data flow, data structures, file structures, and the structure and sequence of screen displays that reflected performance of certain sub-routines.

8. Because the structure of software is an engineering design, holding that a copyright in the program covers the structure was itself a radical step. See, e.g., Samuelson, *Is Copyright Law Steering the Right Course?*, 5 IEEE SOFTWARE 78 (1988). It was unnecessary for the court to go beyond the structure issue and to declare every aspect of programs to be protectable except their general purpose and function.


10. See, e.g., Branscomb, *Who Owns Creativity? Property Rights in the Information Age*, 91 TECH. REV. 38 (1988) (discussing the Lotus lawsuit against competitors with respect to its 1-2-3 program). Apple has also sued Microsoft and Hewlett-Packard over the "look and feel" of its user interfaces.
structures, and although the Whelan decision has met with a virtual avalanche of criticism in the law review literature, the Whelan decision is having some influence on trial court decisions. The Supreme Court’s decision not to hear an appeal in either the Whelan or Plains Cotton cases can only have encouraged companies like Lotus Development and Apple Computer to continue attacking their competitors through copyright litigation.

Significant public policy issues will be decided in the new round of software copyright infringement cases. Unfortunately, judges in software copyright cases have thus far exhibited a disturbing reluctance to look beyond the narrow confines of the dispute between the parties to consider that their broad decisions have the potential to disrupt some positive developments adopted by the software engineering community. These developments were adopted in order to make significant improvements in the productivity of software development projects and to promote standardization in software products that will make them more usable to the public at large.

Judges in recent software copyright cases have rejected out of hand the notion that more “borrowing” ought to be permissible in software cases than in cases involving novels and plays. Relying too heavily on the characterization of software as a “literary work,”

11. Plains Cotton Co-op Ass’n v. Goodpasture Computer Service, Inc., 807 F.2d 1256 (5th Cir.), cert. denied, 108 S. Ct. 80 (1987). Plains Cotton involved a claim that former employees of the plaintiff “took” the design for a software program developed for the plaintiff and used it in making a program for the defendant. Despite similarities in the organization and design of the software, the Fifth Circuit was not persuaded that infringement had occurred and rejected plaintiff’s suggestion to follow Whelan. Id. at 1262.


15. “Literary work” is now defined under the copyright statute in a manner which di-
judges have been blind to the fact that software is a technology and that progress in the field of technological arts may more easily be impeded by strong copyright protection than might be the case in the field of the literary arts.\(^\text{16}\)

It takes only a small quantum of imagination to see why this might be so. Progress in the literary arts is promoted by encouraging the widest possible dissemination of expression.\(^\text{17}\) Also, innovations in the literary arts are not protectable by the patent law. Progress in the technological arts is more often promoted by standardization and adoption of the relatively small numbers of efficient ways to perform certain functions.\(^\text{18}\) In contrast to the literary arts, there is room for patent law to protect certain kinds of software innovations, because they occur in the technological arts, so that copyright law does not

\(^{16}\) There is not yet one software copyright case that speaks of software as a technology. The Third Circuit in Whelan came closest to doing so when it referred to software as a "utilitarian work," but went on to describe an accounting book in Baker v. Selden, 101 U.S. 99 (1879), as also being a "utilitarian work," even though the book merely described Selden's accounting system. Whelan, 797 F.2d at 1235.

\(^{17}\) See Goldstein, supra note 9, at 1122 (noting that copyright seeks to promote ever more diverse expressions).

\(^{18}\) Id. at 1123 (noting that patent law promotes discovery of efficient solutions to technological problems).
have to provide all of the intellectual property protection that software developers need.\textsuperscript{19}

Apart from the general uncertainty about whether the scope of copyright protection for software is or should be narrow, broad, or somewhere in between, there is considerable uncertainty as to the specifics of the software copyright law because there are contradictory judicial decisions about matters of great concern to software developers. Among the unsettled legal issues that show up in the caselaw are: whether the underlying structure of computer programs is protectable by the copyright;\textsuperscript{20} whether user interface formats are part of the program's protectable expression;\textsuperscript{21} whether user modifications to copyrighted software infringe the copyright;\textsuperscript{22} and whether making copies of a program for reverse engineering purposes infringes the copyright.\textsuperscript{23}

Many other significant software copyright issues have yet to be directly addressed by the caselaw. They include: whether and to what extent the functionality of programs is protectable by copyright; whether algorithms are protectable by copyright; what the proper relationship between patent and copyright protections is for software; and who owns intellectual property rights in computer-generated works.\textsuperscript{24}

In traditional copyright theory, it would almost seem absurd to inquire if the functionality of a copyrighted work could be within the scope of the copyright or if an algorithm might be protectable by copyright; the answer to the questions would obviously be "no." Yet, under the two \textit{Whelan} decisions, the answer to both questions would seem to be "yes," functionality and algorithms are protected by copyright.

On the functionality point, the trial judge in \textit{Whelan} said that the manner in which a copyrighted program "operates, controls, and

\textsuperscript{19} See infra notes 61-67 and accompanying text.
\textsuperscript{20} Cf. \textit{Whelan}, 797 F.2d 1222 (protected) and \textit{Plains Cotton}, 807 F.2d 1256 (not protected).
regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information' was part of what Whelan's copyright protected.\textsuperscript{25} The Court of Appeals decision quoted this language with approval.\textsuperscript{26} The appellate court's emphasis on similarities in the ways that certain sub-routines performed, as a basis for its holding on infringement only reinforces the view that it regards functionality as protected by copyright.

The Whelan decision said nothing whatever about the protectability of algorithms under copyright law. However, if one applies the Whelan test for infringement,\textsuperscript{27} algorithms seem to be swept into the fold of "expression." On the other hand, the Whelan test does contain the "no-alternative" limitation that would convert certain aspects of software from the protectable expression category to the unprotectable idea category if it can be established that there is only one or a small number of ways to achieve that function.\textsuperscript{28} Thus, if a particular algorithm is the only one that can be used to implement the overall function of the program, it might be excluded from the protected domain. However, rarely will there be only one or two algorithms that can be used. It will be more common that there will be a limited number of algorithms that can efficiently implement the function. But notice that the Whelan decision does not even restrict the scope of software copyrights to situations in which there is only one or a small number of ways to "efficiently" achieve the function. Why copyright law should be used to deprive software developers of the opportunity of using more inefficient algorithms, structures, or the like, just because one company got to the implementation first, is a mystery.

Even patent law—which normally protects a wide range of things that copyright law considers to be "ideas"—has been slow to protect algorithms. If the Whelan decision is to be taken seriously, it would seem to cover even algorithms, particularly if a software copyright plaintiff's lawyer is astute enough not to call the algorithm an

\textsuperscript{25} 609 F. Supp. at 1320.
\textsuperscript{26} 797 F.2d at 1239.
\textsuperscript{27} Whelan stated that the unprotectable "idea" in software (as if there was always only one idea per program!) is its general purpose or function.
\textsuperscript{28} The Whelan test says that "the idea" (that is, its unprotectable aspect) of a copyrighted computer program is its general purpose or function, while all else about the program is protectable "expression" unless there is only one or a small number of ways to express that "thing," in which case what would otherwise be "expression" becomes part of the "idea," along with the general purpose or function. \textit{See} 797 F.2d at 1235-40.
\textsuperscript{29} The traditional interpretation of Baker v. Selden, 101 U.S. 99 (1879), has been that what patent protects, copyright may not protect. \textit{See} Samuelson, \textit{supra} note 1, at 732-36.
algorithm, but rather to call it the structural backbone of the software. (That, of course, is what an algorithm is.)

Also, it is not settled in software copyright law what procedure should be used for determining when infringement has occurred in software cases and, more importantly, what the test for infringement should be. There is some agreement that the standard Arnstein 30 two-step procedure for making infringement decisions—first, analyzing similarities with the aid of expert testimony and, second, making a "lay observer" impressionistic judgment about whether piracy has occurred—is inappropriate for software.31 The primary reason for modifying the standard approach to copyright infringement analysis is that software is too complex for a lay observer to make an intuitive judgment about infringement, unless there is exact duplication of the code. Thus, experts are needed to make judgments about the degree of similarities and differences.

Moreover, the only thing about a computer program that is easy for judges or juries to make the "lay observer" judgments about, namely the screen displays that some programs generate, can be a misleading basis for an infringement determination. Completely different underlying programs can generate identical screen displays; and the same or substantially similar programs can generate completely different screen displays. Some judges have come to realize that the standard copyright procedure that works for literary and other artistic creations does not work for software. However, the proper role of experts and the proper mix of analysis and impressionistic judgments are far from settled in software cases.

The two major controversies about the proper test for copyright infringement in software cases involve the so-called "look and feel" test 32 (primarily a test pertaining to the user interface of the software) and the "idea-as-general-purpose" test utilized by the Third


31. The only thing on which the Third Circuit in Whelan and the author of the note, Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test, 68 MINN. L. REV. 1264 (1984), could agree was that the Arnstein approach would not work for software. See Whelan, 797 F.2d at 1232. The Third Circuit and the author of the note disagreed as to their respective judgments about how the modification of the traditional copyright test should affect the scope of protection. The note argues that more borrowing should be tolerated in computer program cases. The Third Circuit panel disagreed. Id., at 1238-39.

Circuit in the *Whelan* decision (which would cover all aspects of the software).\(^8\)

The "look and feel" test seems to be derived from the *Roth Greeting Cards* case, in which the court, apparently unable to articulate precisely what the defendant in that case had taken from the plaintiff, decided that it was the "total concept and feel"\(^4\) of the two works that was substantially similar. One can understand why Lotus and Apple might want to substitute "look and feel" for "total concept and feel" as a focus of their user interface litigations, in view of the fact that the copyright statute specifically states that "concepts" (total or otherwise) are not protectable by copyright.\(^5\) "Look" does seem a safer word, though it has the same virtue for plaintiffs as the "total concept and feel" test: a vagueness about what might be within its scope. Whatever legitimacy a "look and feel" approach might have in cases involving artistic or fanciful\(^6\) works (and even this is questionable), it seems singularly inappropriate for cases involving functional aspects of computer screen displays (such as formats for or listings of available commands for the execution of program functions). If such displays can be protected, the next thing you know, judges in software copyright cases will be saying that programs cannot have the same "compilation" of functions as an existing copyrighted program, or that all of the command functions have to be given different names. Such a renaming will make it impossible for a program user to take advantage of the experience he or she may have had with one program and carry it over when using other programs.\(^7\)

As mentioned above, the *Whelan* "idea-as-general-purpose" test has been much criticized, although few of the commentators have a narrower test to offer as an alternative. Several other proposals for tests for software copyright infringement are similar to *Whelan* in being

\(\textit{33. See supra} \textit{note} \textit{27}.\)

\(\textit{34. Roth Greeting Cards v. United Card Co., 429 F.2d 1106, 1110 (9th Cir. 1970). Both the Broderbund and Sofiklone decisions adopt the Roth "total concept and feel" approach.}\)

\(\textit{35. 17 U.S.C. §102(b).}\)

\(\textit{36. Roth Greeting Cards itself is an artistic expression case, as is Sid \& Marty Krofft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977) which endorsed Roth Greeting Cards. See Cooling Systems \& Flexibles v. Stuart Radiator, Inc., 777 F.2d 485 (9th Cir. 1985) (rejecting a "total concept and feel" approach in a copyright infringement claim involving a nonfanciful work such as a radiator catalog).}\)

\(\textit{37. The Lotus complaint, for example, claims copyright infringement of its 1-2-3 program in part because the defendants use the same command names. But see Synercom Technology, Inc. v. University Computing Co. 462 F. Supp. 1003 (N.D. Tex. 1978) (receptive to the argument that user interest in standardization of data input formats was relevant to the scope of copyright protection for the program and its user manual).}\)
overbroad and out of keeping with copyright tradition. One of these proposals, for instance, would focus attention on whether the defendant ever looked at or drew anything from the plaintiff's software in preparing his or her own product. If so, that second product would be tainted, irrespective of how different the two products might have ultimately become.

There is no dearth of law review commentary on software copyright infringement cases, tests, and issues. There is, in fact, almost too much of it. Articles go off in so many different directions that law review commentary, rather than clarifying the law as their authors no doubt hoped would be the case, may have simply added to the cacophony and high level of confusion about software copyright protection that currently exists.

The volatility of software intellectual property law is almost frightening. Take the controversy about "shrink-wrap" licenses. They purport to deprive a software purchaser of any property interest in the software he or she has spent good money to acquire and to deprive him or her of rights to make adaptations or back-up copies that he or she would otherwise have under copyright law. When a federal judge in Louisiana ruled that federal copyright law pre-empted portions of the Louisiana shrink-wrap enforcement law, the outlook for the enforceability of shrink-wrap licenses darkened considerably. The outlook would have brightened if the Supreme Court had decided to review this shrink-wrap case.


39. Conley & Bryan, A Unifying Theory for the Litigation of Computer Software Cases, 6 Computer/Law J. 55 (1985) (formulating a test focusing on the "fairness" of the defendant's conduct, not on the extent of similarities between the works). A rule, such as that proposed by these authors, is particularly inappropriate for technologies, since, for the most part, technological progress happens when technologists "build upon" the work of others. For the same reason, the patent law allows "improvements" to be separately patentable, even if the inventor of the improvement is different from that of the underlying technology, and leaves modest incremental improvements in technology in the public domain.

40. Whether "shrink-wrap" licenses are enforceable as a matter of contract law is a related problem. See, e.g., Hazen, Contract Principles as a Guide for Protecting Intellectual Property Rights in Computer Software: The Limits of Copyright Protection, the Evolving Concept of Derivative Works, and the Proper Limits of Licensing Arrangements, 20 U.C. Davis L. Rev. 105 (1986) (arguing that such licenses are invalid).


There is, to sum up, virtually nothing solid to teach about American software copyright law. When a whole field of law is a welter of confusion and contradiction, it is no small challenge to teach the law as it truly is and keep students' attention and respect. No wonder copyright professors tend to be reluctant to give the subject more than one or two classes.

Apart from the considerable uncertainty of the law, teaching American software copyright law is perilous because even when judges reach sound conclusions in software cases the decisions, more often than not, are either so cryptic as to be uninstructive or so lengthy and flawed as to be painful for both the instructor and students to dissect them with the analytic scalpel of copyright law.

The Plains Cotton decision is a good example of the "too cryptic" variety. What, for example, did the Fifth Circuit mean when it said that "market factors" may require cotton marketing software to be structured similarly? Was it talking about screen displays? Was it talking about the underlying structure of the program? How necessary was it to use the same structure? Why was it necessary? Does this emphasis on the design constraints imposed by "market factors" mean that although the Fifth Circuit explicitly rejected the Whelan approach, it was really following it, because the court identified a situation in which the Whelan "necessity" test for merger of idea and expression had been met?

43. Q-Co. Industries, 625 F.Supp. 608 and E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485 (D. Minn. 1985) are examples of cases that persuasively explain why the judges ruled as they did on the copyright infringement claims.
44. See, e.g., Plains Cotton, 807 F.2d. 1256.
45. My own favorite example of the "right-outcome-for-the-wrong-reasons" phenomenon is Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), cert. denied, 464 U.S. 1033 (1984), analyzed at length in Samuelson, supra note 1, at 742-49. A better and simpler justification for the outcome in Franklin would have been this: In 1980, Congress passed a law admitting computer programs to the copyright system. Congress intended by this action to provide a remedy against those who commercially distributed unauthorized copies of computer programs. Apple developed these computer programs and copyrighted them. The Franklin programs were virtually identical, unauthorized copies of the Apple programs. Therefore, this is the kind of conduct Congress intended to reach by passing the software amendments. (The Franklin argument was a statutory, not a constitutional one.) The Third Circuit's actual opinion in Franklin contains many conceptual errors which blossom in the court's later Whelan decision.

Others prefer the Whelan decision as an example of a right conclusion for the wrong reasons. See, e.g., Raskind, The Uncertain Case for Special Legislation Protecting Computer Software, 47 U. Pitt. L. Rev. 1131, 1161-64 (1986). I argue that Whelan was wrongly decided on the merits and was poorly reasoned. See Samuelson, supra note 8.
46. Plains Cotton, 807 F.2d at 1262.
47. See, e.g., Note, 33 PATENT, COPYRIGHT, AND TRADEMARK J. (BNA) 307 (1987) (speculating that Whelan and Plains Cotton are reconcilable on this basis).
The Whelan decision is perhaps the best illustration of the "over-elaborate" variety. The conceptual flaws in Whelan are so deep and extensive that it is difficult to keep any criticism of it brief. Suffice to say that the judges in that case took the word "idea" too literally. They neglected to consider the other words of exclusion from copyright protection that section 102(b) sets forth.48

It is not just "ideas" in the sense of abstract concepts (such as a program's general purpose or function), but also processes, procedures, methods of operation and systems that copyright does not reach. One reason copyright law does not protect "processes" is that, historically, patent law has been the body of intellectual property law that has done so.49 How is one to separate the program's potentially patentable process from its potentially copyrightable structure? The test for software copyright infringement should contain a sorting mechanism to define which software components patent or copyright law can protect, and which components should be in the public domain.50 It is ironic that the Third Circuit thought that it derived its outrageously overbroad test from Baker v. Selden.51 That decision could have been more properly read to support the "code only" position.52

Apart from the perils caused by the magnitude of uncertainty about the state of the law and the generally unsatisfactory rationales and analyses given in software copyright decisions, there is yet another source of peril for those who teach American software copyright law. In the United States, particularly among law students, there is a strong desire to protect intellectual property. This attitude puts an instructor who raises doubts about the soundness of certain court decisions or legislative judgments which protect software, in peril of being regarded as an intellectual terrorist. Software is perceived as an important source of the future economic strength of America. People want to give software "adequate" protection. Many seem to equate ever stronger copyright protection for software with a perpetually vigorous United States software industry, and excoriate those who question this equation. However, very "strong" copyright protection for software may end up strangling or stifling innovation in the United States software industry instead of invigorating it.

It should have been obvious that when Congress decided to put

50. See Samuelson, supra note 8.
51. Whelan, 797 F.2d at 1235-36.
52. See, e.g., Karjala, supra note 5.
software into its copyright system—a body of law whose most fundamental tenet is the non-protection of technologies—that there would be considerable difficulty in integrating software into that system. Apparently, this was not so obvious at the time Congress passed the software amendments in response to the CONTU recommendations. Only five and a half years after the passage of the software amendments, a study by the Office of Technology Assessment of the Congress concluded that copyright law cannot successfully be applied to computer software. In hindsight, it is easy to see that CONTU was unduly optimistic about how well software would fit into the copyright system.

Now that we have begun to discern how difficult the process of integrating software into the copyright system can be, the problem arises of finding the time and energy to invest in learning enough about software technology to be able to make intelligent judgments concerning the legal issues. Determining the appropriate scope of protection for software under copyright law and how copyright doctrine might be “tailored” to accommodate this strange hybrid—both machine and writing at once—is a significant challenge. It is hard work to try to think through the proper roles for copyright and patent protection for software and to articulate their implications for traditional doctrines and principles of these previously separate and exclusive bodies of law.

The task of tailoring the law is made more difficult, because copyright and patent practitioners and theorists tend to ignore one another's domains. People in the copyright field tend to know the copyright system very well and to think about specific issues exclusively in copyright terms. They tend to have only a vague understanding of the patent system, and are not particularly eager to learn more. The converse is true about patent experts. As a result, copyright lawyers and teachers find it natural to think about software copyright problems in terms of the fine arts: hence the constant allusions in software copyright cases to software being a “literary work.” Thus,

53. Baker v. Selden, 101 U.S. 99 (1879), is generally regarded as the source of the rule that technologies are not protectable by copyright. See Samuelson, supra note 1, at 732-41.

54. See, NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1978). See also, Samuelson, supra note 1, at 665-66 and 692-705, for a critical history of CONTU and the software amendments to the copyright statute.

55. OFFICE OF TECHNOLOGY ASSESSMENT, U.S. CONG., INTELLECTUAL PROPERTY RIGHTS IN AN AGE OF ELECTRONICS AND INFORMATION 81 (1986).

56. This is Menell's term. See Menell, Tailoring Legal Protection for Computer Software, 39 STAN. L. REV. 1329 (1987). He is not the only one who calls for some “tailoring” of copyright law to make it more appropriate for software. See also Raskind, supra note 12.
copyright experts find it natural to refer to cases involving novels and dramatic plays, instead of engineering drawings, when analyzing software copyright problems. Engineering drawings are a more appropriate analogy since the process of developing software design is fundamentally an engineering process.

Often knowing little about technological subjects, which they are willing leave to patent lawyers, copyright lawyers have made an exception for software. But they still find it difficult to think about software protection issues in terms of their technological dimensions. They still want to analyze software problems as if software were a "literary work." They also resist carving out a domain of software protection in patent law. Some copyright proponents even support the idea of concurrent copyright and patent protection for software—that is, that the very same features about software can be protected both by copyright and patent. This is a notion that cannot be supported under traditional copyright and patent theory.

Patent law got off to a slow start as a form of intellectual property protection of software. The Patent Office itself was, in the early days, not particularly receptive to software invention applications. Early Supreme Court decisions raised serious doubts about the patentability of certain software inventions. Indeed, CONTU's argument for copyright protection rested in part on its perception that patent protection was largely unavailable for software. It was not until the early 1980's that the tide turned for software inventors. The Supreme Court decision in Diamond v. Diehr signaled a new era for software patents.

Both the Patent & Trademark Office and the Court of Appeals that reviews Patent Office decisions seem increasingly eager to interpret patent doctrines favorably for software inventions. Patent lawyers are now advising clients to seek patent protection for things that some years before would have seemed out of the question. As more

57. See supra note 15.
59. The author has attended conferences at which software copyright lawyers have made such assertions.
60. See Samuelson, supra note 1, at 727-53.
64. 450 U.S. 175 (1981).
65. See, e.g., Chisum, supra note 61, at 1020 and Appendix.
66. See, e.g., Smith, Yoches & Anzalone, Computer Program Patents, 5 COMPUTER LAWYER 1
and more software patents issue for high level software structures, algorithms and user interface designs, patent law is coming to be a very important source of intellectual property protection for software technology.

It is easy to see that pro-patent and pro-copyright forces are on a collision course. What are American intellectual property law professors doing about it? Some of us talk about the problems. Some of us even write about software issues. But it is a very hard and frustrating set of problems to think about, and it is not made easier or less frustrating by the fact there really is not much that we law professors can do to straighten things out, at least not in the short run.

Ever an optimist and ever a believer in the powers of rationality, I think that things will sort themselves out in time, and that we will eventually look back on this period of confusion comfortably, forgetting how much confusion there really was and how hard it was to envision a better way of doing things. Until then, we have our work cut out for us.

(1988) and Sumner and Lundberg, The Versatility of Software Patent Protection: From Subroutines to Look and Feel, 3 COMPUTER LAWYER 6 (1986).

67. It seems it is still necessary to claim that one's algorithm is a non-mathematical one, even though any computer scientist can tell you that there is no such thing as a non-mathematical algorithm. See, e.g., White and Redano, Patent Opportunities for Software-Related Subject Matter, 4 COMPUTER LAWYER 13, 15 (1987) (discussing the Patent Office guidelines for computer-related inventions).