UIC John Marshall Journal of Information Technology & Privacy Law

Volume 17
Issue 1 *Journal of Computer & Information Law - Fall 1998*

Article 3

Fall 1998

The Relative Roles of Patent and Copyright in the Protection of Computer Programs, 17 J. Marshall J. Computer & Info. L. 41 (1998)

Dennis S. Karjala

THE RELATIVE ROLES OF PATENT AND COPYRIGHT IN THE PROTECTION OF COMPUTER PROGRAMS

by DENNIS S. KARJALA[†]

I. INTRODUCTION

I am delighted to have this opportunity to participate in a symposium devoted to resolving the problem of software patents. I come at the problem from the other side of that broad intellectual property river, namely, copyright, where the debate over the appropriate scope of protection for computer programs continues to rage. This debate has largely followed the copyright tradition of considering copyright as a stand-alone statute, capable of resolving all of its problems with very little reference to other legal regimes and, in particular, with very little reference to patent law. I have the strong impression that patent law has also gone about its business relating to software, from *Benson*[1] to *Beauregard*,[2] as if patent law were essentially the only relevant player. My primary point is that we must start trying to coordinate between these two branches of intellectual property law in trying to find the optimal social policy balance of legal protection for computer software.

For traditional copyright and patent subject matter, these insular attitudes were generally workable because there was little subject matter overlap. With only a few exceptions, copyright has always eschewed

† Dennis S. Karjala is a Professor of Law at the Arizona State University College of Law, where he has taught since 1978. He received a B.S.E. degree from Princeton University in 1961, an M.S. and Ph. D. from the University of Illinois of 1963 and 1965 in electrical engineering, and a J.D. in 1972 from the University of California at Berkeley, where he was Editor-in-Chief of the California Law Review. His primary field for research and writing is copyright law, especially the copyright protection of computer software. He is also leading the academics in opposing legislative proposals to extend the term of copyright protection. For the academic year 1997 through 1998 he is Irving Younger Visiting Professor of Law at the University of Minnesota. The author gratefully acknowledges a writing grant from the Oracle Corporation.

1. Gottschalk v. Benson, 409 U.S. 63 (1972).
2. In re Beauregard, 53 F.3d 1583 (Fed. Cir. 1995).

protection of *functional* works, while functional works of technology are the very stuff of patent subject matter. Moreover, patent law has tried to steer clear of works that are merely useful in the sense of informing human beings or portraying an appearance but not functional in the sense of actually doing work in the physical world. As a result, traditional patent and copyright subject matter divided rather cleanly between the nonfunctional, reserved to copyright, and the functional, reserved to patent. As long as that distinction was largely observed, as it was until the advent of computer programs, there was very little need for communication between the two disciplines.

Computer programs, and in particular the congressional decision to include them among the categories of copyright subject matter, force a rethinking of the old division of labor, because computer programs are functional in a way that traditional law reserved to patent subject matter. Computer programs may, of course, be read by trained people for no purpose other than to understand how they work.[3] A program so used would serve the nonfunctional purpose of informing a human being, a traditional ground for its inclusion among copyright subject matter. However, the primary purpose for which computer programs are written, produced, and distributed, is not that they be read by human beings.[4] Rather computer programs are produced for the purely functional purpose of causing a computer to perform a particular task or set of tasks. Computer programs are, in short, technology—the technology for using computers. They, quite literally, sequentially set switches inside the machine in such a way that the results can be interpreted by human beings as "information processing."

Although few seemed to realize it at the time, the decision to bring computer programs, works of technology and thus functional subject matter, under copyright was a radical break from intellectual property tradition. It occurred partially because the two fields operated independently for so long that most of the participants forgot the essence of the distinction between patent and copyright subject matter. Inability in both camps to separate the form of a computer program from its substance led to further confusion. The copyright cases concerning computer programs exhibit a chronic failure to distinguish merely "useful" works under copyright (like dictionaries and maps) from truly "functional" works, like lawn mowers and, now, computer programs themselves. Consequently, judges in the early software copyright cases made the fundamental mistake of treating computer programs as literary

3. A machine like an automobile engine can be "read" or studied for the same purpose, but we do not on that ground include it within copyright subject matter.

4. In fact, most program copyright owners would prefer to prevent others from reading their programs, if they could do so through legal restrictions or technological means. That way they could protect their copyright-unprotected trade secrets and know how.

works amenable to the same analysis they had used for years in application to works like novels and plays. The result was broad protection of technological features of programs for the very long period of copyright without any showing that the features in question would have qualified for even a twenty year patent. Similarly, judges in patent cases, and the United States Patent and Trademark Office ("PTO") itself, had great difficulty extricating themselves from the form in which this new technology appeared, namely, as written sets of statements and instructions that seemed too far removed from the nuts, bolts, and sockets of the traditional mechanical and electrical products they were used to (and too similar to the "printed matter" for which they had long denied patentability). As a result, we saw some 20 years of § 101[5] subject matter metaphysics in which the basic question of did the claimed invention evince a new and nonobvious technological advance was rarely asked.

Developments in the last few years, and in particular the PTO's Guidelines[6] (*Guidelines*), have begun a new and welcome approach from the patent side. Because the subject matter question has now been mooted, essentially by fiat, patent lawyers must now begin to deal with the real issues of program novelty and nonobviousness. These developments come not a moment too soon. Unless and until patent law gets its act together to protect noncode technological innovation in computer programs, clearly and unambiguously, the pressure to effect such protection under copyright will continue. Copyright protection for most program innovation would be much worse than whatever comes out of patent, because copyright was not designed for, and indeed is ill-suited to, the protection of technology. On the other hand, when patent law *does* accept its traditional role as the primary source of intellectual property protection for program technology (especially noncode technology), copyright can and should feel free to retreat to its primary role as protector of nonfunctional cultural works and its new, important, but definitely secondary role in the protection of technology, namely, the protection of program code from misappropriation.

The articles major subtheme, therefore, is that patent and copyright lawyers must begin talking to each other now that computer programs have brought a degree of overlap to their heretofore largely distinct subject matter areas. I begin by reviewing previous work I have done, outlining the respective subject matter areas of copyright and patent and explaining why we arrived at the traditional division of labor based on the distinction between functional and nonfunctional works. I then summarize my theory for the copyright protection of software, stepping

5. 35 U.S.C. § 101 (1994).

6. Examination Guidelines for Computer-Related Inventions, 61 Fed. Reg. 7478 (1996) [hereinafter "Guidelines"].

through the various elements of computer programs that have been the subject of so much heated debate in the copyright camp but almost no debate among the patent lawyers, namely, program code, program structure or SSO ("structure, sequence, and organization"), and program interfaces. Finally, I take on the question of the proper application of patent law to these program elements.

II. BACKGROUND

A. COPYRIGHT AND PATENT[7]

Why do we have two statutes, patent and copyright, instead of just one? Both protect the fruits of intellectual creativity, and neither demands or protects, in general, more creativity than the other. No one has ever shown that the two statutory regimes protect qualitatively different creativity. Many copyright-protected works are quite mundane, and certainly some patented inventions, like the transistor, can only be said to reflect creative genius of the first order. Moreover, the underlying social policy goals of patent and copyright are similar. Both seek to draw a balance between, on the one hand, providing an incentive for the creation of works desired by society and recognizing in some fair and just way the efforts of their creators and, on the other hand, insuring a broad public domain that permits later inventors and authors to build on the existing foundation to advance technology and culture for the overall benefit of society.

Yet these two statutory schemes for the protection of the fruits of intellectual creativity are radically different from each other and effect their social policy balances in very different ways. Patents must be narrowly claimed and protection is narrowly limited to the claim. Patents must be approved by administrative authorities, must involve a "nonobvious" and not simply a normal engineering advance, and remain valid

7. This and the following section are based on Dennis S. Karjala, *A Coherent Theory for the Copyright Protection of Computer Software and Recent Judicial Interpretations*, 66 U. CIN. L. Rev. 53 (symposium issue 1997) [hereinafter *A Coherent Theory*]; earlier work in which I presented similar arguments include Dennis S. Karjala, *Misappropriation as a Third Intellectual Property Paradigm*, 94 COLUM. L. REV. 2594 (1994); Dennis S. Karjala, *Copyright Protection of Computer Software, Reverse Engineering,* (on file with the author) *and, Professor Miller*, 19 U. DAYTON L. REV. 975 (1994) (this article was originally published under the erroneous title *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*) [hereinafter *Reverse Engineering and Professor Miller*]; Dennis S. Karjala, *Recent United States and International Developments in Software Protection*, 16 EUR. INTELL. PROP. REV. 13 (Part 1) & 58 (Part 2) (1994); Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33 (1987) [hereinafter *New Protectionism*]. On the specific question of the distinction between patent and copyright subject matter, *see also* Amicus Curiae Brief for Respondent, Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996).

for twenty years from filing. Copyrights, on the other hand, come into existence automatically, with no requirement that the rightholder specify precisely which aspects of her work are protected and which are not. The work need only be "original" in the sense of being the intellectual work product of the author and not "novel" or "new" in any absolute sense. The scope of protection can be very vague, with infringement determined by a standard of "substantial similarity."[8] Moreover, the copyright-protected work need only be the intellectual product of its author, with at most minimal artistic "creativity," and protection continues for roughly seventy five to one hundred years. In sum, patents are more narrowly defined than copyrights, harder to get, persist for a shorter period, and are easier to defend against in cases of nonliteral copying.

Why are there these differences when the statutes are aimed at achieving the same overall goal? I suggest that the answer is to be found in the nature of the respective subject matter to which each of these statutes, until the advent of computer programs, was largely confined. As briefly outlined in the introduction, the fundamental difference between traditional patent and copyright subject matter is best captured by the term "functionality." Patents protect creative, *functional* invention; copyright protects creative, *nonfunctional* authorship. In order for the term "functionality" to serve as the benchmark for distinguishing between patent and copyright subject matter, however, it is crucial to avoid equating the term "functional" with "useful."[9] To identify "functionality"

8. In a private communication, Richard H. Stern has presented me with the argument that patents may be broader than copyrights in their scope, given the abstract nature of "algorithms" and similar patent subject matter and given the doctrine of equivalents. I agree that patents do cover some of the abstractions that are excluded from copyright protection under 17 U.S.C. § 102(b) (1994) of the Copyright Act. However, that does not necessarily make the scope of patents *broader*. The scope of a patent is still strictly limited by the claim language. It is hard to imagine, for example, film production machinery or processes infringing a patent on book production machinery or processes, while films infringe book copyrights all the time (if not used with permission). In any event, none of the conclusions reached in this section or in this article as a whole depends on whether patent scope as a philosophical matter is broader or narrower than copyright scope. The real question is why we have differences between the statutes at all, and I stand by my answer to that question, namely, that patent is designed to deal with functional subject matter and copyrights with nonfunctional subject matter.

9. Copyright lawyers seeking to expand copyright protection, especially for computer software, conflate these terms all the time, in an effort to show that functionality is not a bar to copyright protection. *E.g.*, Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUDIES 321, 323-24 (1995) (pointing to copyright protection for dictionaries, maps, and charts and for the output of new technologies, such as photographs); Arthur W. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 986 (1993) (pointing to factual compilations, dictionaries, code books, encyclopedias, advertising, and instruction manuals). I am indebted to Professor Jim Chen for pointing out to me that part of the problem in trying to distinguish between

with "usefulness" leaves no distinction between patent and copyright subject matter and raises the question of why we should have two very different statutory schemes for protecting the same thing.

Copyright protects many works that are "useful" to human beings. Maps enable us to go from one place to another; recipes tell us how to bake cakes; works of art make our walls more attractive, accounting books explain how to implement a particular system of accounting. All of these works are copyright protected. There is a subcategory of "usefulness," however, that has generally been excluded from copyright protection. This category is most aptly captured by the definition of a "useful article" in the Copyright Act:

A "useful article" is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. . .[10]

Although this definition first entered the statute in the Copyright Act of 1976 and certainly not with computer programs in mind,[11] it captures much of the essence of the traditional exclusions from copyright subject matter, following the great case of *Baker v. Selden.*[12]

Thus, a work is "functional" for the purpose of distinguishing between patent and copyright subject matter if it performs some utilitarian (or useful) task *other than* to inform, entertain, or portray an appearance to human beings.[13] "Usefulness" in the ordinary sense (that is, without

"functional" and "useful" for subject matter purposes is actual employment of the term "useful" in 35 U.S.C. § 101 (1994) of the Patent Act.

10. 17 U.S.C. § 101 (1994) (defining "useful article").

11. The Copyright Act applies the "useful article" definition only to one class of work, namely, pictorial, graphic, and sculptural works, which is the category in which traditional works of industrial design would be classified for copyright purposes.

12. Baker v. Selden, 101 U.S. 99, 105 (1879) ("The description of the art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself. The object of the one is explanation; the object of the other is use. The former may be secured by copyright. The latter can only be secured, if it can be secured at all, by letters-patent.")

13. "Functionality" as so defined does not exhaust the distinction between patent and copyright subject matter. A "system," for example, is excluded from copyright protection under § 102(b) (1994), although it may be patent subject matter as a "process." Many systems or processes are conceptual algorithms that inform human beings how to do something and are not self-executing. They are, then, not directly "functional" within the definition offered in the text. "Processes," of course, are explicitly patent subject matter, so it would not be too much of a strain simply to define such processes to be "functional" to maintain the formal distinction between copyright and patent subject matter under a single term. *Cf.* Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091, 1145-46, 1200 (1995) (descriptively applying the label "functional" to systems, processes, and methods of operation to the extent they are excluded from copyright protection under 17 U.S.C. § 102(b)). Attempting the same with regard to "systems," however, is not so easy, because of patents traditional exclusion for "business methods" and "printed matter" (such

the qualifier excluding the conveyance of information or the portrayal of an appearance) does not make a work functional under this definition. A map is nonfunctional, even though it is often quite useful in traveling from A to B, because it does no more than convey information. A doll or toy airplane is nonfunctional even though either may be useful in keeping children productively occupied, because it only portrays an appearance. A picture is nonfunctional for the same reason, even though it may be useful for decorating a house or office. None of these things do more than simply inform or portray an appearance to human beings. A building, on the other hand, is functional, notwithstanding that part of its design portrays an esthetic appearance to human beings, because it also protects from the wind and rain. A computer program, at least in object code form, is functional because it causes a computing machine to operate so as to achieve a certain result. All program-to-program interfaces and protocols are functional under this definition, in view of their inherent utilitarian function of allowing interoperability between computers and the programs that govern computer operations. Many computer program user interfaces, such as the Lotus 1-2-3 user interface at issue in *Lotus Development Corp. v. Borland*,[14] are also functional in the sense used here, because they have the intrinsic utilitarian function, or at least intended function, of permitting users to engage in such operations as inputting and manipulating data in a fast, efficient, and easy-to-master manner. The enormous differences between the patent and copyright protection schemes are largely attributable to patent's primary role as the protector of functional works, as here defined, and copyright's primary role as the protector of nonfunctional works.[15]

as forms for recording or displaying information). For the purposes of this symposium, it is sufficient to note that computer programs are functional in a way that is common to much of traditional patent subject matter and that distinguishes them from traditional copyright subject matter.

14. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court,* 516 U.S. 233 (1996).

15. The Federal Circuit's important decision in *In re* Lowry, 32 F.3d 1579 (Fed. Cir. 1994), recognized as patent subject matter "an efficient, flexible method of organizing stored data in a computer memory." *Id.* at 1580. (quoting from *In re* Bernhart, 417 F.2d 1395, 1399 (C.C.P.A. 1969), the court disallowed a "printed matter" rejection where the invention required that information be processed by a machine rather than by humans). This is precisely the distinction proffered herein for distinguishing patent from copyright subject matter. The *Lowry* opinion also shows the difficulty of relying on terms like "expressiveness" or "creativity" in making the patent/copyright breakdown. The court notes that the prior art disclosed data models that were either "functionally expressive" or "structurally expressive" and that Lowry's invention sought "to optimize both structural and functional expressiveness." *Id.* at 1580. Thus, it is insufficient simply to conclude that something is "expressive" to make it copyright subject matter. Much patent subject matter can also be labeled "expressive" or "creative." *Id.*

The policy basis for these differences between the two protection schemes is the social desirability, indeed necessity, of allowing later technological creators to make incremental improvements on the works of others.[16] The social utility of allowing subsequent authors to make minor variations on a copyright-protected novel, for example, is minimal, so the "substantial similarity" standard for infringement forces authors to write whole new books rather than merely tinker with old ones. Technological improvements, however, are often substantially similar to the products they improve and would infringe if the copyright paradigm were applied. Many such improvements on existing products are rather straightforward, or "obvious" in the sense of patent law, and they are given no intellectual property protection once they are released to the public. While such products often show at least as much intellectual creativity as many copyright-protected works, their creators have a monopoly only for the period that is required for competitors to recognize the value or popularity of the improved product, figure out its "secret," if any, and gear up for production and marketing. In the case of technological products, we have drawn the social policy balance at a different point than for traditional works of authorship, because we believe that to grant intellectual property rights in ordinary engineering advances would hinder the development of more products than it would encourage. Hence the "nonobviousness" requirement of patent law, as well as its shorter term and its requirement for an explicit statement of the claimed invention.

B. Patent and Copyright Distinctions

The functionality/nonfunctionality distinction between patent and copyright has not been historically perfect[17] even before computer pro-

16. *See New Protectionism*, *supra* note 7, at 39.

17. Over the years courts have occasionally afforded copyright protection to isolated examples of functional works, in the sense used herein, although these courts rarely evince an understanding that they have actually brought functionality under copyright. A clear example is that of standardized test questions that seek to measure intellectual or psychological traits from human responses to the questions. *See* Educational Testing Services v. Katzman, 793 F.2d 533 (3d Cir. 1986); Applied Innovations, Inc. v. Regents of the Univ. of Minn., 876 F.2d 626 (8th Cir. 1989). Even those skeptical of the value of some of these tests will agree that their purpose is to measure real-world phenomena, and therefore they have a utilitarian purpose (measurement) other than to entertain or inform. Consequently, they are functional in the sense used in this article. Some blank forms and classification schemes also receive copyright protection notwithstanding their functionality as systems for presenting information in a convenient or otherwise more useful form. Key Pubs., Inc. v. Chinatown Today Pub. Enter., Inc., 945 F.2d 509 (2d Cir. 1991) (yellow page classification scheme); Kregos v. Associated Press, 937 F.2d 700 (2d Cir. 1991) (template for showing baseball statistics). *See generally* Dennis S. Karjala, *Copyright and Misappropriation*, 17 U. Dayton L. Rev. 885, 922-26 (1992). But protecting functionality is not the only mistake courts sometimes make in applying copyright to specific fact patterns. A model for mana-

grams and now, architectural works[18] were taken under copyright's protective wing. It is surely correct to say, however, that copyright has in the main eschewed protection of function in the sense here defined, and judicial and legislative efforts to allow copyright to control markets for functional products have almost always been met with fierce debate.[19] Patent and copyright evolved alongside one another over a period of several hundred years, and yet the protection schemes are very different. There must be *some* reason other than inertia that the two have coexisted so long. Only functionality explains this dichotomy in a fundamental way. If the distinction between the two primary intellectual property regimes based on functionality is not quite descriptively correct, the aberrant cases are better attributed to the long-standing habit of patent and copyright lawyers to stick to their own fields, where they never had to articulate why there was another statute out there doing apparently the same thing. One would expect the occasional error to creep in around the edges when the subject matter boundaries are only implicit. I am aware of no court or commentator who has articulated any standard for distinguishing copyright and patent subject matter that comes anywhere near functionality in its power to explain why we continue to have two statutes instead of one.[20] Therefore, I offer as a normative proposi-

gerial decisionmaking for use in training managers is not directly functional, but it is the kind of thing that should be excluded from copyright protection under Baker v. Selden, 101 U.S. 99 (1880) and 17 U.S.C. § 102(b) as a system or process. Nevertheless, such a system was held copyright-protectible in Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527 (5th Cir. 1994).

18. The recent addition of a separate class of "architectural works" to the statutory list of copyright-protected works means that building designs are no longer subject to the separability test for industrial designs and other "pictorial, graphic, and sculptural" works. Presumably, the long-standing doctrine of *Baker v. Selden* will continue to limit or deny copyright protection to functional aspects of buildings, but only time and the courts will tell us.

19. Professor Reichman describes at length the debates over the inclusion of industrial design within copyright. J.H. Reichman, *Design Protection in Domestic and Foreign Copyright Law: From the Berne Revision of 1948 to the Copyright Act of 1976*, 1983 DUKE L.J. 1143.

20. Professor Friedman has argued that the patent/copyright boundary is largely determined by whether copying is easy and is easily recognized and independent invention is unlikely. David D. Friedman, *Standards as Intellectual Property: An Economic Approach*, 19 U. DAYTON L. REV. 1109, 1118 (1994). For a comparison of his approach and that of distinguishing between the two subject-matter areas based on functionality. *See A Coherent Theory*, *supra* note 7, at 56 n.3. Professor Lunney has expressly rejected functionality as the crucial distinction between patent and copyright subject matter. Glynn S. Lunney, *Lotus v. Borland*: *Copyright and Computer Programs*, 70 TUL. L. REV. 2397, 2420 n.70 (1996). In criticizing his analysis, I incorrectly stated that he did not attempt to explain why we have two very different statutes for the protection of the fruits of intellectual creativity. *A Coherent Theory, supra* note 7 at 65 n.38. Professor Lunney has properly chastised me for this error, for which I now publicly apologize. Indeed, later in his article Professor Lunney supplies an entire section entitled "The Line between Patent and Copy-

tion that copyright, with its low threshold of eligibility, vague infringement standard, and long protection period, should not be allowed to trench upon the traditional domain of patent to protect functional aspects of works absent a clearly articulated social policy basis.

In fact, there exists a social policy basis for protecting computer programs, notwithstanding their functional nature, under copyright. That policy, however, also clearly indicates how copyright protection in programs and their interfaces should be limited to avoid upsetting too much the careful balance between copyright and patent that has evolved over time. The next section summarizes this analysis and the conclusions that follow from it.

III. ANALYSIS

A. The Proper Role of Copyright in the Protection of Software

How did functional works like computer programs get under copyright? The short answer is that copying of *code*, including routine code that is ineligible for patent protection, is fast, cheap, and easy, even though its initial creation can be expensive and time-consuming. In other words, code is vulnerable to a type of misappropriation to which most other industrial products are not. If a competitor copies a nonpatented machine part, he must still tool up for its production, test its quality, and market his product. A competitor who legally copies a computer program, however, can get into production immediately and can avoid all development and testing costs. One can prattle on for as long as one

right." Lunney, *supra* note 20, at 2427-2431. Here he argues that the distinction arises from the practical differences involved in copying the creativity embodied in the two types of subject matter. This conclusion seems similar to that of Professor Friedman, although the analysis is considerably different. Even Professor Lunney's analysis concedes, however, that utility served traditionally as an effective proxy for the underlying policy, which he takes to be preventing competitors from obtaining an unfair commercial advantage through copying. *Id.* at 2430. He points out that the advent of computer programs has reduced the accuracy of the traditional line based on utility in implementing the basic policy of promoting fair and prohibiting unfair competition. *Id.* at 2431. Of course, he gets no disagreement from this author on that score. In fact, it is not clear to me that the differences in our analyses are great. Professor Lunney says that the basic policy is the prevention of unfair competition and that functionality, while serviceable in the past, is not up to the job in the digital age. I say that the traditional distinctions are likely there for some good reason, but that unfair competition possibilities in the digital age force us to make some changes (such as protecting unpatentable computer programs from misappropriation). In any event, we seem to come essentially to the same conclusions on the important issues of copyright protection for programs. For example, he would limit copyright to prohibiting competitors from engaging in mechanical and near exact duplication. *Id.* at 2431. While I am not sure that I would go that far with respect to many traditional works, I have long advocated this result for computer programs. *Id.* He also agrees that Borland's copying of the Lotus command structure should not be considered copyright infringement. *Id.* at 2435.

pleases about the intellectual creativity that is involved in software creation and how well it fits within the formal definition of "literary work" in the Copyright Act.[21] The facts remain that creativity does not and cannot distinguish between patent and copyright subject matter, that computer programs are functional and under the traditional division of labor would have been protected solely under patent and trade secret law, and that formal inclusion of programs as literary works under copyright says nothing about how *much* copyright protection they should have. I have spelled out in detail many times elsewhere in this argument that it was the vulnerability of program code to fast and inexpensive takings that distinguished program technology from all others and justified its being brought under copyright, so I will not repeat it all here.[22]

Program code's peculiar vulnerability to cheap and accurate copying governs the software protection policy goal under copyright. We should protect against piracy—methods of copying that too greatly upset the traditional balances of legal and nonlegal protection available for works of technology—without revamping our entire scheme of intellectual property protection for functional works or for technological creativity.[23] This, at least, is the conservative approach, the approach that least disrupts the traditional intellectual property protection balances, especially the delicate balance between copyright and patent.

Implementation of this policy leads to the conclusion that the program copyright should protect only the literal code and mechanical or electronic translations of code. Higher level aspects of program structure and design should be considered patent subject matter and protectable under patent law if they meet the stricter patent standards. Fanciful aspects of user interfaces, such as video game characters, should be copyright protected as traditional pictorial or graphic works, independent of the copyright in the program that generates them on a screen or other output device. Other aspects of user interfaces must be protected, if at all, either by trade secret law, by an independent copyright as a pictorial,

21. "'Literary works' are works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia . . ." 17 U.S.C. § 101 (1994) (defining "literary works").

22. *See, e.g.*, *A Coherent Theory*, *supra* note 7, at 66-72.

23. This policy goal is virtually identical to the "market failure" analysis of the authors of the *Software Manifesto.* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994). Those authors advocate achieving the policy goal of avoiding market failure (or "piracy" or "misappropriation," as I phrase it) by a sui generis statute, primarily because they believe that elements of programs other than literal code are also vulnerable to incentive-eroding copying. I remain unconvinced that anyone has made the case that noncode aspects of programs are substantially more vulnerable to piracy than any other types of technological products—which remain protected only by patent and trade secret law. Unless and until that case is made, a proper interpretation of copyright and the other branches of traditional intellectual property law can achieve an essentially socially optimal result without additional legislation.

graphic, audiovisual, musical, literary, or similar *traditional* work (rather than as a nonliteral element of the computer program), or as a nonobvious technological advance under the patent standards. These conclusions are developed in the remainder of this article. The rest of this Section explains these conclusions with respect to copyright.

1. *Program Code and Copyright*

Under the United States Copyright Act, a computer program is "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[24] This definition quite explicitly distinguishes between the "set of statements or instructions" constituting the "computer program" and the "certain result" that the program brings about via execution by the computer. The natural reading of this definition leads to the conclusion that the computer program is, quite simply, the code. This straightforward reading, which has been almost wholly ignored by copyright courts caught in the web of reasoning by analogy to traditional copyright cases, has rather important consequences for the scope of protection afforded by the program copyright to so-called "nonliteral elements" of the program and to the protection of program interfaces. For the time being, however, we restrict ourselves to code.

The statutory definitions make clear that code is a "computer program" and, therefore, a "literary work" under copyright. On the other hand, it is equally clear that program code, at least object code, is functional and therefore technological in nature. Electronically stored object code directs the internal operations of a computer without human intervention (except for data input). As a functional work, the traditional functionality limitations of *Baker v. Selden*[25] and the "process" and "method of operation" exclusions of § 102(b)[26] would deny copyright protection. Nevertheless, most code is also routine, in the sense that it represents a straightforward application of standard programming principles and practice to cause a computer to operate to solve a particular problem. Such code would not qualify for patent protection because it is "obvious." Without copyright protection, these programs would be wholly unprotected once widely distributed, and the notion of a computer program copyright would be largely devoid of meaning. This *reductio ad absurdum* argument indicates a congressional intent both that code be protected by the program copyright, subject to merger, fair use, and similar copyright limitations, and that neither *Baker v. Selden* nor § 102(b) should be applied to exclude code from copyright protection. Otherwise,

24. 17 U.S.C. § 101 (1994) (defining "computer program").
25. Baker v. Sheldon, 101 U.S. 99 (1880).
26. 17 U.S.C. § 102(b) (1994).

there would be no copyright protection for code, contrary to obvious congressional intent. Thus, at least to the extent of code, we must infer that Congress recognized implicit exceptions to these fundamental and longstanding copyright exclusions. The whole game under copyright is to decide whether these exceptions for code should extend to other, noncode program elements, such as interfaces or "nonliteral" elements of program structure or organization (often referred to as "structure, sequence, and organization," or "SSO").

2. *Program SSO and Copyright*

The copyright question is usually posed as whether SSO is protected as a "nonliteral element" of the protected program code. The analogy is to the copyright in a novel or play, which has long been recognized to extend beyond the verbatim language to more or less detailed elements of plot sequence. Most of the proponents of broad program copyrights conveniently forget that copyright in other types of literary works, such as histories, fact works, rule books, and technical works, is much "thinner," limited to verbatim language and close paraphrases thereof.[27] They also usually forget that computer programs are only literary works in form; in substance they are the technology for using computers. Consequently, reasoning by analogy to traditional (nonfunctional) works without resort to policy cannot be expected to lead to sensible results, unless one is a strong believer in luck.

Nobody seriously disputes that SSO is functional. A program is not structured for its esthetic appearance to human beings; rather, it is structured in a particular way for the purpose of making it operate optimally under whatever constraints are imposed—such as to be faster, more accurate, easier to repair, or to use fewer scarce or expensive resources.[28] The question then becomes, given that copyright protects code

27. *Compare* Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936), *cert. denied*, 298 U.S. 669 (1936) (play) *with* Feist Pubs., Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282 (1991) (factual compilation); Landsberg v. Scrabble Crossword Game Players, Inc., 736 F.2d 485 (9th Cir. 1984) (game strategy); Miller v. Universal Studios, Inc., 650 F.2d 1365 (5th Cir. 1981) (history); Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 980 (2d Cir. 1980), *cert. denied*, 449 U.S. 841 (1980) (historical theory); Rosemont Enter., Inc. v. Random House, Inc., 366 F.2d 303 (2d Cir. 1966), *cert. denied*, 385 U.S. 1009 (1967) (biography); Continental Cas. Co. v. Beardsley, 253 F.2d 720 (2d Cir. 1958), *cert. denied*, 358 U.S. 816 (1958) (legal form).

28. Even the notorious *Whelan* case recognized this point:

> A program's efficiency depends in large part on the arrangements of its modules and subroutines; although two programs could produce the same result, one might be more efficient because of different internal arrangements of modules and subroutines. Because efficiency is a prime concern in computer programs (an efficient program being obviously more valuable than a comparatively inefficient one), the arrangement of modules and subroutines is a critical factor for any programmer.

Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1230 (3d Cir. 1986).

functionality, should it also protect SSO functionality? Congress has not addressed the question, so we might begin by looking to underlying policy, especially the policy that patent is the norm for intellectual property protection of functional (technological) subject matter.

If we recall that we brought functional program code under copyright because of code's peculiar vulnerability to misappropriation, we can ask whether program SSO functionality is similarly vulnerable. This is not the forum to attempt a detailed answer, but at a minimum it should be obvious that program structure is much more difficult to extract and understand for use in a competing product than the literal code itself, especially if source code is not readily available. Moreover, even when a skilled programmer has learned the new and better SSO, he or she must write copyright noninfringing code to effect the desired result of program execution and must then further test, debug, and market. No one has shown that these "reverse engineering" steps leave computer programs any more vulnerable to misappropriative copying than any other technological product (given that code is protected against copying).

There is, therefore, no policy reason to protect program SSO functionality under copyright. Moreover, while Congress has directed the inclusion of computer programs under copyright, there is no indication one way or another that Congress intended program copyright protection to extend to program SSO. Indeed, standard application of the functionality doctrine of *Baker v. Selden* and the "system," "process," and "method of operation" exclusions of § 102(b) argue to the contrary, until Congress becomes more explicit that it intends to carve out another exception to these fundamental exclusions. Consequently, there are no compelling statutory or technical reasons for including SSO under the copyright umbrella.

The copyright case law, via *Computer Associates v. Altai*,[29] has effectively reached this result by filtering out of the copyright infringement analysis program elements determined by "efficiency." Unfortunately, most judges do not yet realize the logical implications of *Computer Associates* and continue to go through its complex "abstraction, filtration, comparison" process, vainly seeking to resolve, without a concrete standard, the metaphysical question of separating "expression" from "idea" at each level of abstraction they find in a program. In fact, a court that applies the *Computer Associates* filters honestly will soon realize that *everything* in the SSO is present for the purpose of making the program function better, that is, for efficiency reasons. Consequently, under *Computer Associates*, after filtering for efficiency there is very little, if anything, to protect besides the code.[30]

29. Computer Assoc. v. Altai, 982 F.2d 693, 707 (2d Cir. 1992).
30. See *A Coherent Theory*, *supra* note 7, at 77-83.

3. *Interfaces and Copyright*

The copyright protection of computer program interfaces, and user interfaces in particular, requires special attention. Nearly all courts dealing with the problem under copyright to date, including courts that reach conclusions comporting with the analysis propounded herein, have treated interfaces simply as a "nonliteral element" of the computer program generating (or implementing) the interface. This treatment assumes away the fundamental question of whether a particular interface, as such, constitutes copyright subject matter. In fact, the definition of a computer program in the Copyright Act clearly distinguishes between the set of statements or instructions constituting the program and the "certain results" that are achieved by execution of those statements and instructions.[31] All program interfaces, moreover, are a crucial part of what the actual program *does*, that is, a part of the "certain result" that operation of the program brings about. The code is written in such a way that, when executed, the computer presents the interface to the outside world (which includes, besides human operators, other programs and hardware). While the program runs, the computer responds as designed to input signals and produces output signals reflecting the end product of its programmed operations.[32] Consequently, under the statutory definition, interfaces are *not* covered by the program copyright. If interfaces are to be copyright protected, therefore, they must somehow qualify independently as copyright subject matter.

In fact, some aspects of user interfaces may constitute copyright subject matter, such as screen displays (graphic or pictorial works) or "Help" text (literary works). These would be copyright subject matter even if programs were not a part of the copyright picture at all. However, as traditional works, copyright-protected interface elements are subject to the traditional copyright limitations and exclusions, such as merger and, most importantly, § 102(b)[33] as well as the functionality exclusion of *Baker v. Selden*.[34] Congress has implicitly eliminated the application of both *Baker* and § 102(b) to program code, but there is no reason to assume that Congress intended special treatment for traditional categories of works just because they are produced by execution of program code instead of by traditional implements like typewriters or paint brushes.

It will be objected that all this theory is well and good but the courts have already gone down a different path, treating interfaces as nonliteral elements of copyright-protected computer programs. The response is that many courts are getting the correct answers, anyway, if not for pre-

31. *Id.*
32. *See A Coherent Theory, supra* note 7, *at* 96-99.
33. 17 U.S.C. § 102(b) (1994).
34. Baker v. Seldon, 101 U.S. 99 (1879).

cisely the right reasons. Thus, the First Circuit in *Lotus v. Borland*[35] properly denied copyright protection for functional elements of a user interface as an unprotected "method of operation" under § 102(b) without addressing the subject-matter question. The Tenth Circuit in *Mite, Inc. v. IQTel, Inc.*[36] at least got the right answer by denying similar protection on the ground that the challenged elements of the interface were determined by standard programming conventions or dictated by external functionality and compatibility requirements. Consequently, it remains only to supply the right reasons for these denials of copyright protection to bring some coherence between doctrine and holdings.

As a policy matter, there is no reason to protect functional aspects of interfaces under copyright. This is particularly true if these elements are treated as patent subject matter, so that nonobvious advances in the technology of interface design can seek protection for specifically claimed inventions by that traditional route. The Eleventh Circuit has explicitly recognized the importance of separating between program elements protected by patent law and elements protected by copyright:

> It is particularly important to exclude methods of operation and processes from the scope of copyright in computer programs because much of the contents of computer programs is patentable. Were we to permit an author to claim copyright protection for those elements of the work that should be the province of patent law, we would be undermining the competitive principles that are fundamental to the patent system.[37]

Sooner or later, courts are likely to understand the full meaning of this message, at least if patent law recognizes that it, too, has an important role to play in the overall scheme of software protection. If patent law tries to dodge the issue of protecting functional elements of interfaces, however, we can expect courts worried about misappropriation of perceived creative invention to apply copyright to the problem. The result would be broader protection of interface technology than patent would offer, for extremely long periods, and without any demonstration to an expert examiner (or anybody else) that the technological advance is even new, let alone nonobvious.[38]

35. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

36. Mitel, Inc. v. IQTel, Inc., 124 F.3d 1366 (10th Cir. 1997).

37. Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1541 n.21 (11th Cir. 1996). Later the court said, "In no case . . . should copyright protection be extended to functional results obtained when program instructions are executed and such results are processes of the type better left to patent and trade secret protection." *Id.* at 1547 n.33. *See also* MiTek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1556 n.19 (11th Cir. 1996) (cautioning against recognizing copyright protection for user interface elements that constitute a process and therefore patent subject matter).

38. Recall the language of Baker v. Selden, 101 U.S. 99, 102 (1879):

B. The Proper Role of Patent in the Protection of Software

My goal here is to describe a normative standard of what patent law *should* protect when it is faced with claimed technological advances in the art of computer software. The case law has been very adequately explained and analyzed by other contributors to this symposium, and it is not necessary to reiterate much of that here.[39] Nor will I attempt to deal with technical fine points, such as the different enforcement possibilities between apparatus and process patents.[40] I believe that it is first necessary to establish the normative framework. Once that framework is clear, we can turn to implementation details, such as revising the *Guidelines*[41] or amending the statute.[42]

1. *Distinction between Computer Program Innovation and Computer-Related Inventions*

The real open issue with which we should be wrestling now is the patent protection of computer *programs* as such. Claims like those in *Diamond v. Diehr*[43] raise only the issue of *computer-related* inventions, that is, inventions in which a programmed computer is used as a part of some larger system or device. Computer-related inventions, like that in-

> To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the PTO before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.

Id.

39. Another excellent analysis of the development of software patents is Julie E. Cohen *supra* note 9, at 1152-63.

40. Professor Thomas's contribution to this symposium cogently makes the point that this distinction is now largely a matter of drafting in any event. To the extent he is correct for computer program technology, the case is all the stronger for deciding the policy issue of what *should* be protected by patent law, and to what extent, before delving into the technical details to determine whether those results obtain under current law and practice or whether some modifications are necessary.

41. *See* Vincent Chiappetta, *Patentability of Computer Instruction as an "Article of Manufacture:" Software As Such As The Right Stuff*, 17 J. Marshall J. Computer & Info. L. 89 (1998).

42. Lee A. Hollaar, *Justice Douglas Was Right: The Need for Congressional Action on Software Patents*, 24 AIPLA Q.J. 283 (1996)(recommending statutory amendment explicitly treating program implemented processes as statutory subject matter).

43. Diamond v. Diehr, 450 U.S. 175 (1981). *In re* Alappat, 33 F.3d 1526 (Fed. Cir. 1994), also involved a claim to the use of some algorithmic calculation by computer for smoothing lines on oscilloscope screens. The disputed claims did not involve the particular program that implemented the algorithm, and the claims necessarily included limitations defined by the associated hardware and field of use. The facts of Parker v. Flook, 437 U.S. 584 (1978), also place that case in this category, notwithstanding that the Court denied patentability there under 35 U.S.C. § 101 (1994).

volved in *Diehr*, should continue to constitute patent subject matter and will presumably continue to be tested for patentability by the traditional rules. Thus, if there is no claim to a specific program or programming technique but only to an invention that has a "means for" element disclosed in the specification as a programmed computer, the program itself is not the subject of the patent; rather, the programmed computer together with the other elements of the claim are the subject of the patent. If the patent is valid, any computer programmed to do the same thing in conjunction with the other elements will infringe, even if the second program is completely different from the first.

What we need to do now is consider the patentability of the program itself, independent of associated hardware, other software, or field of use limitations. The absence of this issue from the patent cases is probably attributable to their 20-year emphasis on the metaphysical "subject matter" question under 35 U.S.C. § 101. *Beauregard*[44] and the recent *Guidelines*[45] have now largely eliminated this fruitless subject matter inquiry and force us to focus on the actual program. The subject matter question is still with us, but in a more tractable form, namely, does the claimed invention involve technology, as opposed to an "abstract theory" or "law of nature." By "technology" I mean the application of natural laws and the principles of science to accomplish some utilitarian function (useful result in the physical world) other than to inform or portray an appearance to human beings.[46]

Consider a simple example. Suppose the Pythagorean Theorem is newly discovered. Everyone agrees that such a theorem is not patent subject matter because it is "abstract theory" that is not yet applied to the physical world. If someone creates a mechanical device, however, that accepts numerical inputs for the lengths of two sides of a right triangle and grinds out the length of the third side making use of the new formula, the machine, as such, is patent subject matter. The technological advance, if there is one, is in implementing the formula in mechanical elements. If implementing mathematical formulas mechanically is a well established art, no patent would issue for the machine unless the inventor shows a new and nonobvious advance in that art. If a patent issues, its validity should not be affected if someone uses the device other

44. *In re* Beauregard, 53 F.3d 1583 (Fed. Cir. 1995).

45. Guidelines, *supra* note 6.

46. *Id.* For the reasons discussed in the cited note and text, this definition of "technology" fits most comfortably with patent subject matter in the form of machines or articles of manufacture rather than "processes." I do not propose to try to resolve here the question of what processes are technological and what processes are not except insofar as a computer program may be said to be, or to implement, a process. The issue before the symposium is the patentability of computer programs, and this definition of technology suffices to bring the patentable elements of computer programs to the surface.

than to make calculations not directly related to the sides of triangles, and indeed anyone making an equivalent device for some such other purpose would be an infringer. In any event, the theorem itself, being nontechnological, adds nothing to the patent claim for a technological advance. In essence, this is the old "point of novelty" approach to patentability, but addressed under 35 U.S.C. §§ 102 and 103 instead of 35 U.S.C. § 101.[47]

The same analysis can be applied to an electronic device that uses computer software to solve a newly discovered equation or "abstract theory" of mathematics or science. The device (the programmed computer) is a "machine" and, therefore, patent subject matter. The technological advance, if there is one, must lie in implementing the theory in program code. If writing program code is a well established art, as it is, a patent should issue only if the inventor shows that he or she has written code that qualifies as a new and nonobvious advance in that art. No patent should issue if the programmer simply applies straightforward programming principles or practice to the well understood art of solving mathematical problems on computer. That the theory being implemented is new or nonobvious adds nothing to the patent claim for a new and nonobvious advance in computer-use technology. On the other hand, if a pat-

47. I am indebted to a number of exchanges with Professor Chiapetta for allowing me ultimately to focus in on this example and its implications for the patentability analysis. Professor Cohen has earlier made an argument very similar to that presented here, namely, that 35 U.S.C. §§ 102, 103 (1994) of the Patent Act permit the dissection of program-invention claims into statutory and nonstatutory elements in a way that an ordinary subject-matter inquiry cannot achieve. Cohen, *supra* note 13, at 1168-75. She supports the adoption of Richard Stern's "innovative programmer" standard, pursuant to which both mathematical algorithms and general purpose computing equipment are taken as part of the prior art. The nonobviousness issue then becomes whether the claimed invention would have been obvious to a person of ordinary skill who (a) knew the algorithm, (b) desired to accomplish the given task, and (c) desired to do so with the aid of a computer. *Id.* at 1169. Professor Cohen concludes:

> A mathematical algorithm expressed digitally, though not patentable, may be new; general purpose computing equipment is not even that. We are fast approaching an era in which any industrial function can be directed by a general purpose computer with the appropriate software. As with any other useful art, the patent laws should reward only genuinely new and nonobvious advances in the application of computer technology, not the comparatively mundane, though complex, process of adapting a general purpose computer to a particular use with existing programming techniques.

Id. at 1174.

I am indebted to Professor Mark Lemley for reminding me of Professor Cohen's important earlier work on the question here addressed. Professor Moy's contribution to this Symposium argues that if the Patent Office "blue pencils" a nonstatutory part of a hybrid claim but still finds that the claim can surmount the 35 U.S.C. § 101 hurdle, it should not allow later consideration of the excised portions of the claim in making novelty and nonobviousness determinations under 35 U.S.C. §§ 102 and 103. This conclusion too seems similar to those of Professor Cohen and those presented herein.

ent is issued for a new and nonobvious advance in the art of computer programming, anyone else who employs that advance in a computer program, even a program that achieves a different overall result, should be deemed an infringer.

I recognize that real life will necessitate some difficult conceptual line drawing in determining whether a claimant has actually invented new computer-use technology that is patent subject matter or has rather discovered a mathematical idea or "abstract theory" that is not patent subject matter. This general problem is not new to patent law and, indeed, has been a subtheme (sometimes inarticulated) in the program patent cases. However, given that mathematicians solve most formulaic problems by the use of algorithms, and given that computer programs themselves are algorithms for processing the same kinds of symbols that mathematicians have always processed through their "mental steps" algorithms, it may often be difficult to separate the one from the other, especially if they come dressed up in skillfully drafted claim language.

An example based on *Diehr*[48] may be helpful in focusing on the problem. As all patent lawyers know, the claimed invention in *Diehr* involved a rubber molding process that monitored the temperature inside the mold and fed the temperature measurements to a computer for nearly continuous real time recalculation of the cure time for the rubber inside the mold and for a signal to open the mold at the proper time. The computer was programmed to calculate the cure time using the Arrhenius equation, the use of which was not novel in the art of rubber curing. The actual invention, therefore, was continuous monitoring of temperature and real time recalculation of the cure time. This claim, as discussed above, does not involve a claim to the computer program, but only to the process of using a programmed computer in industrial rubber curing.

Suppose now, however, that the process in *Diehr* is in fact one that is well known in the field, including the use of a computer to solve the Arrhenius equation. Our hypothetical invention involves a new algorithm for solving the Arrhenius equation that speeds up the process enormously. This allows more nearly continuous monitoring and recalculation and, let us assume, more accurate gauging of the cure time. Two questions arise: (1) Is this algorithm patent subject matter, and if so it is patentable? (2) Is its programmed implementation patent subject matter, and if so is it patentable? A third question is perhaps whether it is possible to distinguish between the algorithm and its programmed implementation.

The algorithm itself, independent of whether it is implemented on a computer, is a process that calculates numerical results from numerical

48. Diamond v. Diehr, 450 U.S. 175 (1981).

inputs. However, without the computer, there is no application of the algorithm to technology, so we would conclude that it is abstract theory and not patent subject matter, just as we would for newly discovered means of calculating the sides of right triangles or the solution of the quadratic equation.[49]

There is now no subject matter issue in the second question, given that the program is embedded in a computer-readable medium. The harder question of patentability comes down to whether the programmed implementation evinces a novel and nonobvious advance in computer-use technology. Conceptually, the analysis would seem to proceed as follows: to the extent that the algorithm simply serves as the "specification" for the programmer, the algorithm is no more applied technology than a specification to design a computer game with a character that looks like Mickey Mouse. The programmer is told, "Here is the algorithm. Write a computer program that accepts inputs in this form and applies the algorithm to solve the Arrhenius equation." If the programmer applies straightforward programming techniques to solve this admittedly new problem, the program does not result in patentable technology. Not because the program as a whole is not patent subject matter but rather because, as computer-use technology, it contains no patentable invention.

On the other hand, the algorithm may be of a different type. The programmer is told, "Here is the Arrhenius equation. Write a computer program that accepts inputs in this form and solves it as fast as possible." The programmer takes a look at what others have been doing to solve the Arrhenius equation in rubber molding, reads the computer science literature for suggestions on how to solve mathematical equations generally (or even the Arrhenius equation in particular), and writes a program that uses novel and nonobvious programming techniques or methods that cause the computer to solve the equation much faster than it would have under traditional programming techniques. Presumably much of her program will be standard code, but her novel techniques constitute a nonobvious advance either in solving the Arrhenius equation or in solving mathematical equations in general by computer. These

49. On the other hand, algorithmic mathematical models of real life applications of natural laws arguably should be considered processes constituting patent subject matter independent of their implementation as software. Irah Donner, *Patenting Mathematical Algorithms That 'Embrace' Mother Nature*, 9 COMPUTER/L. J., No. 5, at 1, 12-13 (1992). I make no attempt to resolve this problem, because its resolution does not affect the analysis of the proper scope of patent protection for computer programs as such. Professor Chiapetta's paper, as I understand it, recommends modifying the *Guidelines* to insure that this type of claim is made as a process independent of computer implementation. This would help immeasurably in holding the necessary distinctions in mind between computer-use technology and more general processes.

techniques should be patentable, provided she can articulate them in a patent claim that can be understood and implemented by others who wish to write similar programs.

This analysis requires that we distinguish between a computer program implementing an algorithm and the implemented algorithm itself. Only the former is potentially patentable computer-use technology. It seems possible, perhaps even likely, that this distinction will prove to be at least as elusive as the famous copyright distinction between protected "expression" and the unprotected "idea" expressed. Nevertheless, the difficulty of drawing the line in specific copyright cases does not relieve us of the obligation to do so. In copyright, the idea/expression distinction has long since ceased to have much analytical power and serves primarily as a descriptive general rule for separating the things that courts have determined on policy grounds to be protected or not protected by a copyright. But for the vast run of cases, we know fairly well where we are, and new cases are decided by analogical reasoning to prior cases dealing with similar subject matter.

Presumably, a similar development will be necessary in the patent protection of programs. In this respect, however, patent law has some important advantages over copyright, which gives hope that the period of confusion will not last too long. Patent law requires a precise statement of the claimed invention, which is a concept almost entirely foreign to copyright.[50] If the novel and nonobvious techniques that are claimed as the invention are only useful when implemented by computer, the claim will have to include the computer as a limitation.[51] As long as such a limitation is a part of the claim, there is no danger of the patent's "preempting the algorithm." Indeed, depending on the nature of the invention and the state of the prior art, the claim may have to be limited to the field of rubber curing. With time, the PTO and the courts should be able to develop fairly reliable indicia for distinguishing between the aspects of potentially patentable computer-implemented algorithms that truly are computer-function enhancing as opposed to clever but unpatentable new ways to break down the mathematical formula into pieces that enhance our ability to solve it independent of computer assistance.

The following subsection attempts to make the analysis of programs as computer-use technology more concrete by considering the kinds of

50. A few copyright software cases have suggested that courts could, and even should, require the copyright owner to specify the original and non-17 U.S.C. § 102(b) (1994)-excluded elements that she deems to have been infringed. *E.g.*, MiTek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1555 (11th Cir. 1996). Other cases are discussed in *A Coherent Theory*, *supra* note 7, at 109 n.211.

51. Again, Professor Chiapetta's paper seems to come to essentially the same conclusion, but with slightly different reasoning. *Id.*

things that have been coming up for years now in the copyright cases: code, SSO, and interfaces.

2. *Computer Programs and Patent Law*

A fundamental point bears repetition: computer programs are technology—the technology for using computers to accomplish such tasks as word processing, industrial plant management, or scientific research. Computer programs are written for the purpose of causing a general purpose computer to take on specific characteristics for interacting with human operators, with hardware, and with other software. Together with inputs from these interacting entities or parties, programs cause the computer to perform a particular task or set of tasks. While human beings can read and understand computer programs when written in source code, no one reads code unless it is to understand the program for repair or improvement purposes or to learn how to write better programs. Moreover, while the program may be constructed in part with a view to subsequent need for repair or improvement by persons other than the original designer, this is no different from designing an automobile with a view to easier or less expensive repair when parts fail. The underlying purpose of the program, or the auto, is not to communicate with human beings. Rather, the underlying purpose is to perform nonhuman work, in the case of the computer, in the form of rapidly combined and distributed electronic signals.

Computers, at least at present, do not "read," "interpret," or "understand" computer programs. No combination of anthropomorphic language changes the simple fact that computers process electronic signals according to the laws of physics. Because humans are able to place combinations of these signals in a one-to-one correspondence with binary numbers, and because we have invented logic circuitry that "operates" on binary signal inputs to produce results that mimic those achieved by such human mental operations as addition and subtraction, we interpret the computer's actions as "information processing." The computer, however, is simply doing what comes naturally, that is, what it is forced to do by the laws of nature.

Given, then, that computer programs in general are the technology for implementing tasks on computers, should there be limits on the availability of patent protection for them beyond the requirements of traditional patent law, usefulness, novelty, and nonobviousness? The following subsections address this question separately for program code, SSO, and interfaces.

a. Patent Protection for Code

Program code is computer-use technology in its most basic form. If code is to be denied patent protection, that denial cannot be based on the ground that code is not patent subject matter. Source code represents a process for converting input signals into output signals, both of which signals have real world meaning.[52] Moreover, code stored electronically on a diskette, hard disk, or CD-ROM is literally a machine part. Consider a hypothetical "all-purpose vehicle" that allowed substitution, by insertion in a "slot" under the hood, of different "black boxes" that somehow converted the vehicle from gasoline to diesel, or from automobile to power boat, or (since we're getting whimsical) from automobile to airplane or to typewriter or to video game, depending on the particular black box we insert. Each of these black boxes contain innards constructed of traditional mechanical parts; levers, pulleys, gears, and so forth. No one would assert, I assume, that these black boxes do not constitute patent subject matter. Rather, in determining patentability, we would look at claims directed to the inner workings of the boxes and ask whether a novel and nonobvious advance in the art of constructing such black boxes had been effected. If a particular claim was of sufficient generality that it could improve the operation of many different types of these black boxes, regardless of whether they made the all-purpose vehicle "airplane" or "typewriter," we would not require that the inventor specify any particular field of use to which the invention would be confined; the claim would cover any use of the same invention in a black box to be used in the all-purpose vehicle.

Program object code on a diskette is every bit as much a machine part as these hypothetical black boxes. In its executable form, the code exists as *physical* signals (high or low voltages or magnetic field strengths), not as ones and zeros that may be readable by humans.[53]

52. Professor Hollaar recommends statutory amendment explicitly making processes implemented by computer programs patent subject matter, to avoid the judicial interpretation that a patentable process requires some change of physical state. Hollaar, *supra* note 44, at 297-98. As discussed in the text, the computer program represents a process for converting physical signals from one form to another, and those conversions are useful to human beings in the same way that a mechanical calculator is useful to human beings by converting signals with similar meaning and interpretation. While the latter is technically a "machine," both are equally "useful" to humans in exactly the same way. It simply cannot be correct as a matter of social policy that innovation in the second area is patent subject matter but not the first. I should think that would be sufficient to end the argument of whether this process is statutory, but if statutory amendment is necessary to get to this result, then by all means let us do so.

53. Professor Jim Chen has reminded me that analogy to DNA technology is apt here. A geneticist can "read" a base sequence as CAGGTCA, for example, which is a string of human-understandable symbols (and technically a "literary work" under the Copyright Act). Nature, however, neither knows nor cares about these labels. The actual biochemical

These physical signals may be used directly by the computer in the course of operations. For reasons of operational efficiency, however, these program signals are usually not directly used by the computer but rather are transferred from the diskette to other parts of memory (usually with the intervention of storage on a hard drive), but the transfer is one to one and exact. After the transfer has occurred, the program is again stored as *physical* signals that directly govern computer operations from RAM according the to laws of physics. Consequently, while today the program does not directly govern computer operations from the diskette, its exactly transferred copy does. A computer does not *have* to copy the program from diskette and store it in RAM to operate. It just operates more efficiently if it does so, so it would exalt form over substance to base patentability distinctions on how the hardware interacts with the program instead of on the nature of the program itself.

Therefore, any denial of patent protection for electronic-form object code cannot rest on the absence of direct functionality of code or on a purported distinction between code and traditional patent subject matter. And given that electronic-form object code is simply a physical and obvious manifestation of source code (it is source code run through a compiler), there is no fundamental difference between new and nonobvious object code that makes a computer work better and new and nonobvious source code that, when compiled, makes the same computer work better in exactly the same way. Indeed, because programmers write programs in source code, all of the human "invention" actually occurs at that stage, even though it is physically manifested only in electronic-form object code.

Therefore, this whole subject matter debate for program code never really should have started. The problem for program code under patent law is not that it is not patent subject matter. Rather, even as patent subject matter, programming in its mature state may be a kind of technology in which nonobvious advances are rare. Most programs, although often the result of the input of vast quantities of time and money, are simply straightforward application of well understood computer science principles and techniques.[54] If, as Congress has assumed, programs

processes that take place, whether naturally or in the laboratory, result from the unique *physical* structures of these four molecules. A particular sequence can result in the production of a protein because the sequence presents a specific molecular structure to the cell environment that serves as a physical template for such production. *See generally* Dennis S. Karjala, *A Legal Research Agenda for the Human Genome Initiative*, 32 JURIMETRICS J. 121 (1992 Special Issue). We humans greatly increase our understanding of and ability to manipulate these processes by thinking of the actual molecular sequences as strings of A, C, G, and T bases, but we must be careful not to confuse these abstract labels with the underlying physical reality.

54. At least one commentator has argued that implementing a well defined process in a programming language is *always* obvious and therefore nonpatentable. G. Dukarich, *Pat-*

must be protected from misappropriation in order to encourage the investment needed for their development, patent law does not do the job. That is why we went to copyright for protection of code. The question is whether patent protection for code should also be possible.

The *Guidelines* would apparently protect code as such, provided it is claimed as a "computer readable medium encoded with a computer program."[55] We should ask whether it makes any sense to protect the same thing under both patent and copyright. In fact, patent protection that is truly limited to nonobvious advances in the art of writing specific code in a specific programming language may not be a bad idea. Copyright law's merger doctrine would likely not protect specific code sequences that are uniquely useful or efficient,[56] and this is presumably the kind of code, if it exists at all, that one would have to have in order to show a nonobvious and therefore patentable advance. The Patent Office should recognize patentable code only in rare and unusual circumstances, however. Copyright protects code in general, and we should leave it to copyright to figure out how to draw the appropriate social policy balances between protection and free use, at least absent clearly demonstrated technological invention.

b. *Patent Protection for SSO*

SSO is also technology. No one designs or structures a computer program so that it appeals to the esthetic taste of humans. While some aspects of SSO may be described by professionals as "creative," "beautiful," "elegant," or words of similar import, this terminology only reflects their way of expressing praise at a design that works well to achieve its purpose within the given constraints.[57] A program is structured to make

entability of Dedicated Information Processors and Infringement Protection of Inventions that Use Them, 29 JURIMETRICS J. 135 (1989).

55. *Guidelines supra* note 6, 61 Fed. Reg. at 7482.

56. Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1545 (11th Cir. 1996) emphasized that filtration to eliminate unprotected elements must occur at both literal and nonliteral levels. *See A Coherent Theory, supra* note 7, at 89-90, for a discussion of how to distinguish between protected and unprotected code under copyright. The conclusion there is that functionality, as such, cannot be a bar to copyright, because all code is functional, but that merger should apply in the usual way and that some sort of efficiency filter may also be necessary.

57. *See Reverse Engineering and Professor Miller, supra* note 7, at 997 & n.66. The cited footnote illustrates the point with quotes from a materials science series aired on Public Television, which repeatedly compared materials researchers to artists in the way they work, creating new materials that are "stronger, harder, lighter, and smarter" than any we have seen before. Clearly these new materials are patent subject matter. Yet, the series closes with these words about materials scientists: Without their dreams there is no progress, no true science. We wait and watch as they stretch their imaginations and artistry, like Michelangelo with his marble, Matisse with his cutouts, or Aeschylus with his quill. *Id.* As discussed earlier, imaginative or artistic creativity does not distinguish be-

it work better, where "better" means optimizing the desired characteristics as constrained by the environment in which the program is to operate. That a particular solution may be described by aficionados as "elegant" cannot affect its status as patent subject matter.

Given the role of SSO in determining program quality, there is no reason to deny patent protection, at least as a question of patent subject matter. Of course, now that computer programming is a mature art, there may be few nonobvious new ways of organizing or structuring programs that will qualify. In any event, the claim would have to be directed toward a new and nonobvious way of constructing, structuring, or organizing program elements (including, for example, the construction of data files) that has the useful effect of causing the computer to operate in the claimed (improved or better) manner. New and nonobvious sorting techniques that improve on what programmers were using before should be patentable. Object-oriented programming, if it was the invention of a single person instead of the end result of lots of incremental development, may represent the kind of fundamental development in programming methodology that should have constituted patent subject matter.[58] The data structure in *Lowry*[59] would qualify, at least as far as subject matter is concerned. New programming languages should also be considered candidates for patent protection, at least if their inventor shows that the new language is capable of being compiled for actual use in a computer. A new language, like Java, is a methodology for writing computer programs. Such a language is also the user interface of the program code that serves as its compiler or interpreter. As the next subsection shows, it could also be considered a technological aspect of that program's interface and would also constitute patent subject matter as such.

Since an SSO claim is to a methodology, a way of putting computer programs together, it should not matter whether the program or technique is actually implemented in a working program on a "computer readable" electronic medium or is written out in source code or even described on paper in ordinary English. If the claim is to an article of manufacture as object code on a diskette or similar medium, it is still only the new methodology for grouping, arranging, and causing interaction among program elements that is the new and nonobvious advance. Only that advance should be covered by the SSO patent. In this sense, the

tween patent and copyright subject matter. Nor does application of the language of art appreciation to technological advances affect the status of those advances as patent subject matter.

58. This assumes that object-oriented programming, or some similar methodology, can be specified sufficiently concretely to enable others to make use of it in creating actual programs, so that it drops from the realm of abstract idea into implementable technology.

59. *In re* Lowry, 32 F.3d 1579 (Fed. Cir. 1994).

PTO requirement that the program be embedded on some physical medium in executable form reflects a misapprehension of the nature of the invention, notwithstanding that it does at least get us over the incoherent subject matter hurdle that has impeded intelligible discussion of software protection for such a long time.[60]

This view of patentable SSO narrows the patent protection in a program to those elements of structure or organization implemented in the program that are novel and nonobvious advances in the art of programming. While it is unlikely that very many such advances appear in the vast run of popular programs on the market, if such an advance is invented, there is no reason to tie the new methodology to a field of use simply to get over the subject matter barrier. A mechanical or hardwired electrical calculator is patent subject matter, regardless of whether it is used to calculate stock prices, count oranges, or solve differential equations. Similarly, a program on diskette that implements the new methodology may use that methodology for word processing, business operations, or video games. It is the methodology that is the subject of the claim, unless for other reasons the claim is limited to a field of use.

SSO technology easily fits within the traditional conception of patent subject matter, when properly apprehended as the technology for creating computer programs that are themselves the technology for operating computers. Moreover, as a normative policy matter, SSO *should* be considered patent subject matter. If it is not, increasing pressure will be brought to bear to bring copyright into the picture.[61] This will result in protection of *all* original SSO, whether or not nonobvious or novel, under copyright. Such a development could have a large negative impact on the advance of technology. First, the copyright term is seventy

60. Professor Chiapetta's contribution emphasizes the important differences between claiming an invention as an article of manufacture or as a process, and I do not mean to trivialize these differences. Professor Thomas's contribution, however, shows how easy apparatus and process claims can "morph" into one another. If I ignore these differences here, it is to emphasize the primacy of clarifying the basic subject matter question for the various elements of computer programs. Once we agree which of these elements are patent subject matter and which are not, we can set about classifying them as articles of manufacture or processes in ways that seem sensible, perhaps even tweaking the statute in the manner suggested by Professor Hollaar. *See* Hollaar, *supra* note 42. However we end up classifying programs for patent purposes, it should be clear that, while executable code fits comfortably within a general understanding of a machine part, see the discussion *supra* in the text accompanying note 52, the essence of any noncode invention in a program is fundamentally a process or methodology.

61. Hollaar, *supra* note 42, at 289. For many years I and many copyright colleagues such as Professors Pamela Samuelson, Peter Menell, and Jerome Reichman have been strenuously arguing that copyright should not extend too far beyond literal code because of copyright's fundamental misfit with functional works and because patent is potentially available for truly innovative noncode aspects of software. Now it the time for patent law to to back us up!

five (and proposed to go to ninety five) years.[62] Moreover, the copyright standard for infringement is "substantial similarity," so any programmer who has seen another program's structure is prohibited from using it or anything substantially similar to it, regardless (at least arguably) of technological efficiencies and regardless of whether the structure in question makes any advance in SSO technology that would meet the traditional standards for a twenty year patent.[63]

On the other hand, if SSO is patent subject matter, then we need only worry about tying up new and useful SSO for 20 years, that which will be tied up will be specified precisely in the claims, and (at least in principle—but we must encourage the PTO to be vigilant) only significant advances in SSO technology will pass muster. The rest will be free for further incremental improvement by all programmers. These fundamental differences between the way patent and copyright law provide incentives for their respective traditional subject matters exist for good reason. Except for the peculiar vulnerability of program code to misappropriation through copying, programs are no different from any other technological subject matter. Patent law is the field that has been designed to protect technology. Patent law *must*, therefore, take up the cudgel and do its job.

c. *Patent Protection for Interfaces*

Program interfaces require special attention because they can involve both functional and nonfunctional aspects. Interfaces are the doors and windows through which users, other software, or hardware communicate with the program. Interfaces are analogous to lock and key arrangements, nut and bolt arrangements, control panels, and the like. For most programs, interface design is considered separately from the coding, and in that abstract sense the interface has an existence independent of the particular code implementing it.[64] On the other hand, this abstract interface design does not exist in the physical world until program code has been written that defines and implements the inter-

62. While that may seem well beyond the useful life of most programs, we should remember that much mainframe operating software is, or is derivative from, software that is over thirty years old. Moreover, whether it is Microsoft or some other company, it seems likely that somebody will end up with a single dominant operating system for most of the world's PC's. Copyright protection may essentially impede competition in this market in perpetuity.

63. The doctrine of "unconscious copying" means that if a program structure has been discussed in public or published, no subsequent programmer using that or a similar structure can be sure he or she is not infringing.

64. In the *Lotus* case (*See infra,* note 60), Borland implemented the Lotus menu command hierarchy with independently written, and copyright noninfringing, code. Lotus Dev. Corp. v. Borland Int'l Inc., 799 F. Supp. 203 (D. Mass 1992) rev'd, 49 F.3d 807 (1st Cir. 1995), affd. by an equally divided court, 516 U.S. 233 (1996).

face in accordance with the design.[65] Once the interface is implemented in code, the user, whether that be a human being or another computer or program, must use the operational aspects of the interface in the way required by the program or the program will not work properly.

The easiest interfaces to deal with as far as the patent subject matter question is concerned are those that are invisible to the human user, such as communication protocols. As discussed above,[66] unless an interface element can independently qualify under § 102(a) of the Copyright Act, it is not copyright subject matter. Copyright protection for invisible aspects can attach only through a fictitious treatment of the interface as a "nonliteral element" of the program; treatment that cannot withstand analysis under the Copyright Act's definition of a computer program.[67] Moreover, even treated as copyright subject matter, these aspects of program interfaces should be excluded from copyright protection by § 102(b)[68] (as a "process" or "method of operation") and the doctrine of *Baker v. Selden*[69] (requiring intellectual property protection of functional works or systems to be under patent law). This same analysis supports the argument that these invisible aspects of interfaces *are* patent subject matter, a methodology for using computer programs.

Other aspects of screen displays can independently qualify as § 102(a) copyright subject matter, for example, as pictorial works (e.g., video game characters), graphic works (e.g., spreadsheet formats), or literary works (e.g., help screens). These graphic, literary, and pictorial displays produced by programs are traditional subject matter of copyright to the extent that they have no function other than to inform or to portray an appearance. To that extent they are copyright protected if original and, having no functional characteristics of the type that distinguishes patent from copyright subject matter, are not and should not be patent protected regardless of their implementability by a program embedded on a computer readable medium. In other words, the programmed medium may remain statutory subject matter, but no part of any patent claim can validly cover any of these nonfunctional aspects because they are not the means for doing anything, they are not technology.

65. This is similar to any other program function. The basic function of a program, and indeed most of its other specifications, are determined prior to the writing of any code. One writes code to perform a function, such as word processing; one does not just write code willy nilly and then sit back to watch whether perchance it happens to perform the desired function. The function is implemented by the code and exists only as an abstract concept until so implemented.

66. *See* discussion *supra* note 65.

67. *Id.*

68. 17 U.S.C. § 102(b) (1994).

69. Baker v. Seldon, 101 U.S. 99 (1879).

Other aspects of user interfaces are either purely or partly functional. An interface may be designed partly for esthetic reasons but also to allow the user to operate the computer more easily, faster, or with fewer errors or more easily corrected errors. User interfaces may be designed to make operation of the program easier to learn and to remember. The intellectual property issue with respect to such aspects is whether to protect them at all and if so, whether under patent or copyright.

These aspects of interface design are functional if one accepts the findings of psychology and human factor analysis that some of these things work better than others at accomplishing such results as minimizing learning time and errors and maximizing user retention and speed of performance.[70] A problem with treating user interface designs aimed at such functional results as patent subject matter is that, even if they do work better, we often may not know exactly why. This can present problems with defining the patent scope, especially when the doctrine of equivalents is factored into the analysis. Nevertheless, to deny patent protection flatly would be to open the door to copyright protection for *all* original functional aspects of interface design.

There is good authority for denying copyright protection to functional aspects of user interfaces, in both § 102(b)[71] of the Copyright Act and in the more general doctrine of *Baker v. Selden*.[72] Indeed, the Supreme Court in that case explicitly denied copyright protection to the "user interface" involved, a blank accounting form for use in implementing a new system of accounting. The Court did not inquire whether the system, or the implementing form, really *did* work better than anything that had gone before. That job, by implication, is for the patent system. The point is that the system was *intended* to be better in some ways important to the user besides esthetic enjoyment. That was enough to deny the copyright.

Nevertheless, notwithstanding solid grounds for denying copyright protection to functional aspects of interfaces, if the PTO tells the creators of interfaces designed for efficient human-machine interaction that such systems and methods of operation are not patent subject matter, the result will be to leave the creator out in the cold as far as intellectual property protection is concerned. There is substantial room for doubt whether *Baker v. Selden*, venerable though it is, will be sufficiently robust over the long term to hold off the pressure for some form of copyright protection without help from the patent side. If patent law does not

70. Peter S. Menell, *The Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1053-55 (1988).

71. 17 U.S.C. § 102(b).

72. *Baker*, 101 U.S. at 99.

do its share, we should expect judges in copyright cases to try to fill some of the perceived gaps. We have already seen a number of examples in which copyright courts, eager to avoid what appeared to be "piracy" or misappropriation in a particular case, have sought to expand copyright protection for such subject matter.[73]

We will perhaps have to develop patent standards demanding very precise claiming of functional interface innovation, and we might have to limit commensurately the operation of the doctrine of equivalents in this area. However, the difficulties in specifying the claim with precision in the case of functional aspects of user interfaces should not be a ground for flat denial of patentability on subject matter grounds. Otherwise, the courts are likely to develop a form of copyright protection for *all* original aspects of interfaces under copyright, whether or not they represent a

73. Such thinking may have motivated the district court in Lotus Dev. Corp. v. Borland Int'l Inc., 799 F. Supp. 203 (D. Mass. 1992) rev'd, 49 F.3d 807 (1st Cir. 1995), aff'd by an equally divided court, 516 U.S. 233 (1996), and may also have been involved in producing the 4-4 split on the Supreme Court, 116 S. Ct. 804, (1996). While the First Circuit got things basically right in *Lotus*, the Tenth Circuit on essentially the identical issue got to the right result without any reliance on 17 U.S.C. § 102(b) (1994) § 102(b), Baker v. Selden, 101 U.S. 99 (1879) or the relative roles of patent and copyright in interface protection. Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366 (10th Cir. 1997). Even the Eleventh Circuit in Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1541 n.21, 1547 n.33 (11th Cir. 1996), while expressing clear concern about maintaining the border between patent and copyright in terms of functionality, refused to hold that interface specifications are uncopyrightable as a matter of law. *Id.* at 1547. Other copyright cases have protected various kinds of "systems" for presenting information, not obviously different in kind from the blank form denied protection under *Baker*. *E.g.*, American Dental Ass'n v. Delta Dental Plans Associated, 126 F.3d 977 (7th Cir. 1997) (protecting a "taxonomy" of dental procedures); Practice Management Information Corp. v. American Medical Ass'n, 121 F.3d 516 (9th Cir. 1997) (protecting a "coding system" for identifying medical procedures); Key Pubs., Inc. v. Chinatown Today Pub. Ent., 945 F.2d 509 (2d Cir. 1991) (copyright protects the "organizing principle" for yellow page classification, without reference to § 102(b)'s denial of protection for "principles"); Kregos v. Associated Press, 937 F.2d 700 (2d Cir. 1991) (protecting the choice of nine categories of baseball statistics for presentation in tables useful in predicting game results). The reasoning of Judge Easterbrook in the *American Dental Association* case should be particularly a cause of concern for a patent law worried about losing its jurisdiction to copyright. Judge Easterbrook placed heavy reliance on copyright protection of computer programs and standardized test questions, both of which are clearly functional works, to conclude that function is not a bar to copyright protection. The opinion shows no awareness that computer programs represent the first time in history that we have expressly brought a functional work under the copyright wing nor that the reasons for doing so may have nothing to do with other types of functional works that people may seek to protect under copyright. Nor does the *American Dental Association* opinion recognize that, while *Baker v. Selden* and § 102(b) have been partially overruled in the case of program code, they remain in full force with respect to the rest of copyright subject matter. *Id.* Standardized test questions, too, while clearly copyright protected, are arguably a special case. Dennis S. Karjala, *Copyright and Misappropriation*, 17 U. DAYTON L. REV. 885, 922-24 (1992). These developments show how closely copyright is waiting in the wings, ready to spring into action if it appears that patent is allowing too much "creativity" to fall prey to "piracy."

nonobvious advance through software in the usability of screen displays, keyboards, joy sticks, or other devices that we use to interact with a computer. The scope of protection will be defined by copyright's vague substantial similarity standard, so any substantially similar interface will infringe, even though it marks a technological improvement on the old one. The inability to win many "no substantial similarity" defenses on summary judgment will further chill incremental advance in the development of interface technology. No one will want to take a chance if his lawyer tells him that he can know for sure that he has not infringed only after going through a full fledged trial on the merits.

IV. CONCLUSION

Both copyright and patent have important but complementary roles to play in the protection of computer programs. Copyright is wholly unsuited to the protection of most works of technology, but copyright protection of literal program code works well to make unlawful the kind of cheap, fast, and exact reproduction that, if permitted, could easily undercut incentives to produce and distribute computer programs. Moreover, copyright protection of code does not appear to be a great barrier to incremental improvement of most program technology, so at least in general it is plausible that the social benefits from copyright protection of code outweigh the costs of protecting this form of technology even without requiring that the technology be innovative at the level we have always demanded for patents. Copyright, of course, also protects many nonfunctional products of computer programs, such as video game characters, that independently qualify as traditional copyright subject matter. Computer programs and digital technology do not present us with any new conceptual problems in this regard.

Copyright protection for functional but nonliteral program elements, such as SSO and functional aspects of interfaces would represent an unnecessary and indeed unfortunate intrusion into the subject matter of patent law. Patent law can and should protect such functional program elements, provided they meet the standard tests of novelty, usefulness, and nonobviousness. Unless patent law clearly and unambiguously accepts this role, we can expect copyright lawyers to argue, and copyright judges to accept, that copyright protection must be expanded from program code to these nonliteral elements, notwithstanding that functionality disqualifies them as traditional copyright subject matter. The result will be a vast expansion of copyright protection of technology that is likely to inhibit far more technological development, and for a longer period, than society will get back in the form of more and better program technology. The *Guidelines*[74] are an important first step in bringing pat-

74. *Guidelines, supra* note 6.

ent law back into the game as an active player. We need only to clarify that the nature of a patent claim to program technology must be directed to the methodologies for creating and using computer programs and to insist that claimants specify with precision just how their claimed invention qualifies as computer-use technology.