THEORETICAL FOUNDATIONS FOR THE PROTECTION OF COMPUTER PROGRAMS IN DEVELOPING COUNTRIES

Dennis S. Karjala†

TABLE OF CONTENTS

INTRODUCTION

The basic economic argument for some sort of intellectual property law protection of computer software in developing countries has been made elsewhere[1] and need not be repeated here. I accept those general arguments and assume further that

† Professor of Law, Arizona State University College of Law. This article was originally prepared for presentation at the International Conference on Intellectual Property Rights in Computer Software and their Impact on Developing Countries, Indian Institute of Science, Bangalore, August 20-21, 1993.

1. Anthony L. Clapes, The Soft Revolution: Economics, Intellectual Property and Software Creation in Developed and Developing Countries (Aug. 19, 1993) (unpublished manuscript presented at the 1993 International Conference on Intellectual Property Rights in Computer Software and Their Impact on Developing Countries, on file with Dennis S. Karjala).

software creation is an activity that most developing countries will want to encourage. As personal computers become more widely held in these societies, greater demand will arise for a variety of programs that perform the same tasks as those already available in developed countries, only with user interfaces better attuned to local languages and customs. These programs may best be written by local software creators, but few such creators will be willing to release their programs to the public without, at a minimum, protection against the piracy of verbatim electronic copying. Therefore, I address the question not of *whether* computer programs should be protected, but rather of the degree to which and the mode in which they should be protected.

Most of the developed countries have already chosen copyright as the mode for protecting computer software. The ongoing judicial and academic debates center on the scope of copyright protection — what noncode elements, if any, of the program are protected, whether and to what extent the program copyright or an independent copyright covers the user interface, and whether and to what degree reverse analysis of programs should be permitted. I have long argued[2] that copyright, notwithstanding its very long term of protection and its unwillingness to recognize compulsory licensing, *can* serve as a useful vehicle for protecting programs. This is possible if the scope of protection is held appropriately narrow and if programs are considered sui generis works requiring legal interpretations unique to their characteristics as technological rather than artistic products.[3] In this article I conclude that copyright protection of programs can equally meet the needs of developing countries, provided that such limitations are applied to its interpretation.

2. *See* Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33 (1987) [hereinafter *New Protectionism*]; Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part I*, 13 EUR. INTELL. PROP. REV. 195 (1991); Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part II*, 13 EUR. INTELL. PROP. REV. 231 (1991).

3. This is not to deny "artistry" in the creation of programs. There is similar artistry in the creation of many other technological products, however, which have never been copyright protected. Works designed to achieve some functional purpose other than to inform or entertain (portray an appearance to) human beings have traditionally been within the realm of patent or trade secret law but not copyright.

I. THE POLICY BASIS FOR COPYRIGHT PROTECTION OF SOFTWARE[4]

A. The Need for Protection Against Piracy

The two primary branches of intellectual property law in the economically developed countries are patent and copyright. Patents must be narrowly claimed, must be approved by administrative authorities, require a truly "inventive" and not simply a normal engineering advance, and remain valid for fifteen to twenty years. Copyrights, on the other hand, come into existence automatically, with no requirement that the rightholder specify which aspects of her work are protected and which are not. Moreover, the copyright-protected work need only be the intellectual product of its author, with minimal artistic "creativity," and copyright protection continues for roughly seventy-five to one hundred years.

Traditional patent law protected creative functional works, such as works of technology. Traditional copyright law protected creative nonfunctional works, such as works of art, literature, and music. The liberal copyright standards for the protection of creative *authorship* implement different policies than the more restrictive patent standards for the protection of creative *invention* because of the social desirability—indeed necessity—of allowing later technological creators to build upon and improve the earlier works of others.[5]

Technology improves incrementally. The copyright infringement standard of "substantial similarity" is simply not appropriate for technological works, because nearly every improvement on such a product is, in the unimproved portions, substantially similar to the first product. Computer programs, communication protocols, hardware-to-software and software-to-software interfaces,[6] as well as many user interfaces, are intrinsically functional. Why, then, did the developed countries turn to copyright instead of traditional patent law for the protection of computer programs?

---

4. Much of the following section is based on Dennis S. Karjala, *Recent United States and International Developments in Software Protection: Part I*, 16 Eur. Intell. Prop. Rev. 13 (1994), and Dennis S. Karjala, *Recent United States and International Developments in Software Protection: Part II*, 16 Eur. Intell. Prop. Rev. 58 (1994).

5. *New Protectionism*, *supra* note 2.

6. Because user interfaces may include fanciful, nonfunctional aspects (such as a video game), it is often important to distinguish them from other interfaces and communication protocols. However, the term "communication protocols, hardware-to-software and software-to-software interfaces" is long and clumsy. Henceforth I refer to them with the equally clumsy but shorter term "nonuser interfaces."

The superficial answer in the United States is that computer programs fit the formal definition of a "literary work" under the Copyright Act.[7] However, the basis for protecting new technological creations for the long period of copyright must stand on more than technicality. This is especially true for a "substantial similarity" test for infringement that can prohibit building on earlier advances without the showing of creative invention required for even a seventeen-year patent. Indeed, United States courts have traditionally rejected copyright protection for functional works otherwise fitting the copyright definitions.[8]

The real reason the United States adopted a copyright scheme for programs is that many programs, including programs that are costly and time-consuming to develop, are simply the result of technologically straightforward applications of well-known programming principles to well-defined problems. These programs do not meet the requirement of traditional patent law for a non-obvious advance in the art.[9] Yet once these programs

---

7. " 'Literary works' are works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia . . . ." 17 U.S.C.A. § 101 (West Supp. 1994).

8. The basic principle that the copyright reproduction right cannot be used to control unprotected ideas or utilitarian features of functional works, and the long copyright tradition underlying this principle, has been most fully developed and articulated by Professor J. H. Reichman, the United States' leading authority on the national and international protection of industrial designs. *See* J. H. Reichman, *Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research*, 42 VAND. L. REV. 639, 692-93, 693 n.288 (1989) [hereinafter *Applied Scientific Know-How*]. *See also* Brief *Amicus Curae* of Eleven Copyright Law Professors, Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992), *amended by* Order and Amended Opinion, D.C. No. CV-91-3871-BAC, Jan. 6, 1993 (reprinted as *Brief Amicus Curae of Eleven Copyright Law Professors in Sega Enterprises Ltd. v. Accolade, Inc.*, 33 JURIMETRICS J. 147 (1992) [hereinafter *Amicus Brief*]). *See generally* J. H. Reichman, *Goldstein on Copyright Law: A Realist's Approach to a Technological Age*, 43 STAN. L. REV. 943, 970-76 (1991).

The seminal case in the United States is Baker v. Selden, 101 U.S. 99 (1880). *See also* Taylor Instrument Companies v. Fawley-Brost Co., 139 F.2d 98 (7th Cir. 1943), *cert. denied*, 321 U.S. 785 (1944), and Brown Instrument Co. v. Warner, 161 F.2d 910 (D.C. Cir. 1947), *cert. denied*, 332 U.S. 801 (1947) (graphic works denied copyright protection when designed to fit with the indicators on a measuring instrument). *See generally* Pamela Samuelson, *Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of Lotus v. Paperback*, 6 HIGH TECH. L.J. 209, 226, 226 n.73 (1992).

9. At least one commentator has argued that implementing a well-defined process in a programming language is *always* obvious and therefore nonpatentable. Gary Dukarich, *Patentability of Dedicated Information Processors and Infringement Protection of Inventions that Use Them*, 29 JURIMETRICS J. 135, 160 (1989). Moreover, computer programs "as such" are barred from patentability under Article 52 of the European Patent Convention, although it is not entirely clear just what "computer programs as such" means. Jürgen Betten, Patent Protection for Software, Appendix 1 (Apr. 27, 1993) (unpublished manuscript presented at the Brussels ECIS Symposium, An Emerging World-Wide Consensus on Software Protection?, on file with Dennis S. Karjala).

are distributed in object-code form, they can be copied almost costlessly in large numbers. Without some form of protection, we should expect that they would be underproduced. Because the problem was slavish copying (especially slavish electronic copying) and because copyright protects at least against *that*, copyright became a natural candidate for the protection of programs.[10]

Maintenance of perspective is important, however. We must bear in mind that other nonpatented works of technology may be freely copied, modified, and improved, no matter how creative the technology may be. The creators of such technological developments enjoy only the limited monopoly resulting from the lead time their products have in the market before they can be successfully produced by competitors. Such copying is permitted—even applauded[11]—not because society devalues technological creativity, but rather because technology advances incrementally and because forbidding such copying would inhibit more creativity than it would engender. Program code arguably requires a different kind of protection because to allow verbatim copying of programs would reduce even their lead time monopoly almost to zero, and that level of protection seems too little.

The goal of software protection in the developed countries, therefore, should be protection against piracy and not a wholesale revamping of the intellectual property protection scheme for functional works or for technological creativity. Piracy, in this sense, refers to methods of copying that too greatly upset the traditional balances of legal and nonlegal protection available for works of technology. 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. If there are grounds for affording even broader protection to program technology under copyright, these grounds do not arise out of traditional copyright itself, notwithstanding the formal classification of computer programs as literary works, because traditional copyright did not protect functionality. Beyond their vulnerability to piracy, I have yet to hear of *any* argument that convincingly distinguishes computer

10. *See New Protectionism*, *supra* note 2. *See also* Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part I*, 13 EUR. INTELL. PROP. REV. 195 (1991); Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part II*, 13 EUR. INTELL. PROP. REV. 231 (1991). Another advantage of copyright is the immediate international nature of protection under the copyright treaties. Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part I*, 13 EUR. INTELL. PROP. REV. 195, 196 (1991).

11. Thomas M. S. Hemnes, *Three Common Fallacies in the User Interface Copyright Debate*, COMPUTER LAW., Feb. 1990, at 14.

programs from other technological products and leads to a broader scope of copyright protection for them. Because broad copyright protection for functional technology is the radical rather than the conservative deviation from traditional norms, the burden of providing a convincing policy basis for broad copyright protection of functionality should be on those seeking it.

One further point should be made explicit: copyright protection of source and object code, the set of statements or instructions that constitute computer programs, is in itself significant protection. Protection of code makes direct copying either for sale or for simultaneous use by others (for example, within a given business) illegal. Of course, not *all* instances of program copying can be detected, any more than all copying of other types of copyright-protected works can be detected. No matter what the law says, some people will continue to make copies of programs borrowed from friends without paying. Sales of unlawful copies in significant amounts, however, *are* readily detectable, because sellers must advertise. Moreover, any employer who distributes illegally made copies of a program among employees in the business runs a serious risk that at some point a disgruntled employee will report the illegal conduct. Therefore, protection of program code alone is of great importance. It is simply wrong to say, as did a committee report for the European Parliament during the debates on the European Software Directive, that if reverse analysis were permitted, "legal protection for computer programs would virtually cease to exist."[12]

B. The Roles of Patent and Trade Secret

Program code will usually, if not always, be ineligible for patent protection.[13] However, patents are readily available in both the United States and Europe for creative advances in so-called "nonliteral elements" of programs that meet the patent standards.[14] There is no need, on top of this scheme of patent protection, for copyright to protect allegedly "expressive" nonliteral program elements, notwithstanding that such elements (as all inventions) are the product of creative human thought. *Every* such program element is intended to serve the functional goal of

12. Eur. Parl. Doc. 134.405, DOC EN/PA/74903 (Dec. 1989) (Mr. Karel Pinxten Rapporteur), *quoted in* Thomas Vinje, *The Legislative History of the EC Software Directive*, *in* A Handbook to European Software Law 37 (M. Lehmann & C. Tapper eds., 1993). Such a statement not only misses the trees (the effectiveness of legal prohibition of literal code copying in economically important situations); it does not even see the forest (general protection against piracy of code).

13. *See supra* note 9 and accompanying text.

14. *See* Betten, *supra* note 9.

causing a computing machine to operate to achieve a result better, faster, more efficiently, or more easily. Such works should remain within the realm of patent absent a policy basis for more radical change than that required to protect against simple piracy.[15]

Moreover, trade secret law can protect patent- or copyright-unprotected elements of programs that are not widely distributed (i.e., those that are kept secret). Trade secret law is based on the social advantages of more efficient production resulting from assurance to employers that they can share with employees advantageous techniques and information that are not widely known. There is nothing unique to computer programs that would argue for a more limited form of trade secret protection for them than that granted to other trade-secret-protected works. Trade secret can protect such nonliteral elements as algorithms, methods of interoperability with other programs or equipment, and information stored in program files or in separate databases, in addition to program code. Relying solely on trade secret law for protection of nonpatented algorithms, production processes, or valuable data has the advantage that protection extends only so long as the court determines it should take to legitimately reverse-engineer the product or otherwise independently generate the secret.[16] If such program aspects are copyright-protected, however, the doctrine of "unconscious copying" may result in the barring of an employee from working for a competitor on

15. None of this is to say that the Patent Office never makes mistakes. Some argue that patents were issued for algorithms and methods that are obvious and thereby get in the way of technological advance rather than promote it. *E.g.*, Simson L. Garfinkel, Richard M. Stallman, Mitchell Kapor, *Why Patents are Bad for Software*, ISSUES SCI. & TECH., Fall 1991, at 50-51, 53. Mistakes of this type are endemic to the patent system, however, especially when new technologies are involved. In the long run, courts and the Patent Office usually, if clumsily, reach a balance that society can live with. *See generally* Dennis S. Karjala, Thinking Beyond Patents for the Protection of DNA-Sequence-Related Inventions (May 25, 1993) (unpublished manuscript presented at the Workshop on International Cooperation for the Human Genome Project: Legal Aspects, Bilbao, Spain May 24-26, 1993, on file with Dennis S. Karjala). The issue, therefore, is not whether mistakes have been made. Rather, it is whether the current patent system is inherently flawed specifically with respect to computer programs and whether we can come up with something better. To deny patent protection to all nonliteral aspects of programs would likely cause courts to grant broader copyright protection to those aspects, in ad hoc efforts to prevent what they see as "reaping where someone else has sown." Given copyright's long period of protection, that result could be even worse than overly protective mistakes under patent law.

16. *E.g.*, Integrated Cash Management Services, Inc. v. Digital Transactions, Inc., 732 F.Supp. 370, 378 (S.D.N.Y. 1989) (six month injunction chosen with a view to how long it took plaintiff to create its computer system and the need to neutralize the improper "head start" gained by wrongful use of the trade secrets).

projects similar to those in his prior employment for the full term of the copyright.[17]

C. The Proper Role of Copyright in Program Protection

For the United States, Europe, and Japan, I have argued that a correct scheme of copyright protection for programs and user interfaces would protect against methods of copying activity that, if allowed, would undercut incentives to produce socially desirable works. Further, it would not impede standardization or interoperability and would allow later creators to build on and improve the functional (nonpatented) products of others. Finally, it would avoid legal standards requiring expensive and time-consuming litigation over vague factual questions, such as whether a particular block of code is in some sense "efficient." Rather, the protection scheme should focus judicial attention on the fundamental question of whether the defendant's activities afford an unfair competitive advantage. "Unfairness" would be determined by analogy to traditional means of technological competition and not by repeated incantation of copyright terminology like "copy" or "substantial similarity." In other words, copyright protection for programs should really be something of a misappropriation law. It definitely should *not* be a law protective of all "creativity" that a court finds in a program or its interfaces. Much of the creativity in a program and its interfaces enables the program to function better, and intellectual property protection for those aspects is the traditional job of patent law.

17. In Gates Rubber Co. v. Bando American, Inc., 798 F.Supp. 1499 (D. Colo. 1992), literal copying of code was explicitly not found, and the court defined the "central issue" as concerning the "calculation methods" involved. Moreover, the court based its ultimate finding of copyright infringement on "special regard" for the copying of certain mathematical constants developed by the plaintiff for use in its formulas. These holdings seem contrary to section 102(b) of the United States Copyright Act, which denies copyright in any "procedure, process, system, [or] method of operation" as well as a United State Supreme Court decision stating that facts—such as numbers—are not objects of copyright protection. Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1295 (1991). In addition, the finding of copyright infringement in *Gates Rubber* was wholly unnecessary, as the author of defendant's program was a former employee of plaintiff. The court properly concluded that the constants involved were trade secrets that were wrongfully appropriated. Trade secret law is a much more appropriate means of protecting such valuable information, because the normal remedy is an injunction only for such a time as it would take to legitimately reverse engineer or otherwise uncover the information. If such factual information is copyright protected, however, and someone has legitimate access to it, independent creation as opposed to "unconscious copying" can be very difficult to prove. The Tenth Circuit therefore properly reversed the determination of copyright infringement, while affirming the trade secret violation. Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823 (10th Cir. 1993).

More importantly, because program *code* is vulnerable to a form of piracy (immediate, exact, and almost costless copying) to which other functional works are not susceptible, the scheme should prima facie prohibit verbatim or near verbatim copying of substantial blocks of code. That result is clearly accomplished by the current protection of programs under copyright. In the early days of software protection, we might well have, indeed probably should have, stopped there. This would have left all ideas, processes, methods, principles, languages, technologically efficient blocks of code, systems (including user interface systems), and functional aspects of what the program *does* (like sending lock and key signals to other programs or making mathematical calculations) to the realm of patent law. Protection of *code* under copyright and protection of other program elements either as independent copyright-protected works (e.g., the visual appearance of video games on the screen) or under patent or trade secret law continues to serve as a sensible first cut at a division of labor between the various intellectual property regimes. Program languages, nonuser interfaces, and functional aspects of user interfaces (such as those aspects that make them easier to learn or to use) would not be protected by copyright.[18]

18. Because the cases and the commentators often use the term "functional" without definition but with a wide variety of meanings, it is helpful to set out my own definition. In my mind the best starting point is the definition of a "useful article" under the Copyright Act: "A useful article is an article having an intrinsic utilitarian function that is not merely used to portray the appearance of the article or to convey information . . . ." 17 U.S.C.A. § 101 (West Supp. 1994). Anything qualifying as a useful article under this definition should be considered functional. Thus, a video game as it appears on the screen is not a useful article (even though it may be "useful" to parents in keeping their children occupied) because its function is simply to portray its own appearance on the screen. Nor are maps, rule books, or instruction manuals useful articles—notwithstanding that we all find them "useful"—because their sole function is either to portray an appearance or to convey information, or both. Fanciful or decorative aspects of user interfaces are not useful articles because they serve no utilitarian function other than simply to portray themselves (for the admiration or amusement of the human viewer). All other aspects of user interfaces, however, exist for some other purpose—to store numbers in a spreadsheet cell, to move text, to define macros, and so forth—and therefore would be functional under this definition. Otherwise arbitrary choices that serve, or could serve, the utilitarian function (whether or not "intrinsic" to the particular choice) of standardization across users, whether to permit either easier learning of improved programs or easier user switching among programs performing the same tasks, should also be considered functional. Programs generating video games would of course be functional under this definition, although the fanciful aspects of the audiovisual work that appears on the screen would not. Literal program code is also clearly functional under this definition, although it should remain copyright protected. The protection of code, notwithstanding functionality, is precisely the major change from tradition when we protect programs under copyright. The remaining question is whether *more* functionality should be copyright protected and, if so, why.

We should also beware of new electronic methods of non-value-added reproduction of programs or interfaces that substantially reduce the cost of copying or reverse engineering the work in comparison with the cost of its original production. The use of code generators, which generate code to produce a display already shown on the screen (via the target program), to generate new programs seems a likely candidate for prohibition under this reasoning.[19] Absent a reason for believing that some nonliteral element of a program is subject to piracy in this sense or a ground for copyright protection of the element independent of the program copyright, we should treat programs as functional works of technology that may be highly creative but need no special form of intellectual property law protection that is not generally available for other works of technology. In other words, the goal should be to protect against the evil we have identified—code piracy—and, otherwise, to make as little change as possible in the scheme of intellectual property protection that has worked satisfactorily in the past.

Thus, a fundamental shift of emphasis is required in the copyright analysis: instead of looking for copying, which has been the principal concern of traditional copyright, we must look at the *method* by which copying is accomplished.[20] Whether the method results in an unfair competitive advantage and therefore must be prohibited should be measured by a balance of factors such as comparative costs, value added, and the availability of similar copying or reverse-engineering methods for other technologies. This would make program protection under copyright in a sense analogous to trade secret protection, which looks to the methods by which secret information is acquired to determine liability. Copyright would become, to this extent, a misappropriation statute.

The suggested interpretation seeks to maintain the traditional policy balances of intellectual property law. Because we have brought under copyright a functional work of technology for which copyright was never designed, *some* adjustment of traditional copyright principles is necessary. The required shift of emphasis for program protection, however, is not nearly as radical as that effected by some of the early copyright decisions and advocated by the protectionist commentators, which would

19. This was the method used in CMAX/Cleveland, Inc. v. UCR, Inc., 804 F.Supp. 337, 355 (M.D. Ga. 1992).

20. For an argument that it would be socially beneficial to look to unfair methods of copying, rather than copying generally, and to give limited *protection* to other currently unprotected works, like complete electronic databases and digitally formatted public domain texts, see Karjala, *Copyright and Misappropriation*, 17 U. DAYTON L. REV. 885 (1992).

bring under copyright protection many functional aspects of works formerly protected (if at all) only by patent. Moreover, while copyright has not traditionally looked to misappropriation as such to determine infringement, it does in fact protect against many forms of misappropriation through its prohibition against copying.[21] Finally, because the policy basis of the suggested approach is grounded on the sui generis nature of computer programs as copyright-protected works, it runs much less risk of disrupting the analysis of copyright protection in traditional works.

II. DOES THIS APPROACH WORK FOR DEVELOPING COUNTRIES?

I have outlined above a scheme of copyright protection for software that would protect program code, notwithstanding its functionality, under copyright. This scheme leaves protection of other functional aspects of programs or their interfaces to their traditional patron saints—patent and trade secret—absent an affirmative showing that some aspect of the program is vulnerable to a type of piracy to which other technological products are not subject. This is a minimalist interpretation of copyright protection for programs, but it is also the approach that is most closely aligned with traditional intellectual property law. Protection of functional products under copyright, with its long term of protection, is a *radical* change from the traditional division of labor between copyright and patent. With the exception of code, I see no reason for such radical change in the balance between patent and copyright.

Developing countries clearly have a need and desire for technology transfer from developed countries. Permitting outright theft of computer programs, however, is hardly technology transfer in any meaningful sense, because most do not learn technology simply from copying a computer program, especially a program in object-code form. Moreover, if programs are not protected from this type of piracy, fewer local people or firms will have the incentive to develop new programs of their own for distribution, and the technological lag will grow rather than diminish. Consequently, the case for protecting code against literal copying or non-value-added electronic or mechanical translations seems to hold as true for the developing countries as for the developed.

Whether that protection should come from copyright or some sui generis statute is a question that can be legitimately de-

21. *Id.* at 886-87.

bated.[22] Copyright does have the advantage that protection is immediately international via the copyright treaties. Moreover, with most of the economically advanced countries having already joined the copyright camp, it may be a little late for others to try a new direction. *Properly interpreted*, copyright protection for literal code does have the potential to provide the optimal level of protection.

Consideration should be given in the developing countries to the drafting of specific limitations on the scope of copyright protection in programs when and if the copyright statutes are amended to include them as objects of protection. This would give more assurance that courts in those lands will not make the same mistakes found in many early United States decisions that went well beyond the protection of literal code to protect so-called "structure, sequence and organization" and even user interfaces under the program copyright.[23] Japan is an example of an economically advanced country that has included explicit limitations on the scope of copyright protection in programs. These limitations appear well designed to achieve an optimal level of copyright protection.[24]

Assuming that a developing country accepts my fundamental conclusion that literal code should be protected, three important problems should then be addressed: (1) What elements of a program beyond literal code, if any, should be protected? This is the scope-of-protection problem. (2) Should the protective scheme for the program also cover its interfaces, including its user interfaces, and if so to what extent? (3) To what extent, if at all, should reverse analysis be permitted? These are basic substantive problems that must be faced whether or not copyright is chosen as the protection mode. Presumably even a sui generis statute for program protection would look something like copyright, possibly with a shorter protection term and/or some provi-

22. *See* Pamela Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471 (1985).

23. The great offenders are Whelan, Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222 (3rd Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987) (programs), and Lotus Dev. Corp. v. Paperback Software Int'l, 740 F.Supp. 37 (D. Mass. 1990) (user interfaces). Some judges apparently thought their mission was to protect all the creativity they found in programs and user interfaces—creativity whose scope of protection could only be defined through litigation rather than through a well defined patent claim and the protection of which would last seventy-five to one hundred years. The year 1992 showed important retreats from these extreme protectionist positions. *See* Karjala, *supra* note 4.

24. *See* Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part II*, 13 EUR. INTELL. PROP. REV. 231 (1991).

sion for compulsory licensing.[25] If these issues are not faced in the statute, they will inevitably arise and be decided in individual cases through the courts. Therefore, I proceed on the assumption that copyright will be the legal mechanism chosen.

A. Scope of Protection

The scope-of-protection problem has already been discussed at some length. The goal is the protection of programs against literal copying and electronic translations of code. Such protection follows automatically from copyright, because copying is precisely what copyright is designed to protect against. All other so-called "nonliteral aspects" of programs, possibly excepting certain aspects of user interfaces, are purely technological, that is, they exist solely for functional purposes. Absent a clear demonstration of a policy basis for removing such aspects from the patent scheme, we should continue to rely on the balances in intellectual property law that have been honed over the ages.

In theory, the idea/expression distinction of copyright can be applied to achieve this result. All that is needed is judicial recognition that the "abstractions line" between idea and expression should be drawn somewhere not far above literal code. In practice, as the United States experience through most of the 1980s shows, this result will be reached by judges only through years of backtracking and hand waving.[26] Perhaps explicit statutory limi-

25. Few if any commentators have suggested that the formalities required for obtaining a patent would solve many of the problems associated with program protection. Moreover, a higher patent-like threshold for eligibility would defeat the basic purpose of the entire exercise, which is to protect programs, at a minimum, from the piracy of literal copying. *See supra* note 9 and accompanying text.

26. Many of the early scope-of-protection cases, having accepted that programs are "literary works" under the statute, inappropriately analogized them to novels and plays, in which the scope of protection is broad and extends to overall plots and moderately detailed themes. These decisions failed to recognize that, even among literary works, courts have always varied the scope of protection in accordance with the nature of particular classes of works. Protection in histories, biographies, rule books, legal forms, and scientific works is "thin," and only near-verbatim copying infringes. *E.g.*, Landsberg v. Scrabble Crossword Game Players, Inc., 736 F.2d 485, 488 (9th Cir. 1984); Miller v. Universal Studios, Inc., 650 F.2d 1365 (5th Cir. 1981); Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 980 (2nd Cir. 1980), *cert. denied*, 449 U.S. 841; Continental Casualty Co. v. Beardsley, 253 F.2d 702, 706 (2nd Cir. 1958), *cert. denied*, 358 U.S. 816 (1958). While computer programs are not really analogous to *any* traditional literary work—they may be literary works in form but in substance they exist solely to perform a task on a machine—if the analogy to traditional works is to be made, it would seem that scientific works or rule books would be more appropriate than novels and plays. Consequently, there is broad judicial authority for limiting the scope of protection to literal or near-literal code under traditional law.

tations on the scope of program protection, similar to those in the Japanese Copyright Law,[27] would be advisable.

B. PROTECTION OF INTERFACES[28]

1. Nonuser Interfaces

Interfaces are, by definition, the doors and windows through which a computer program communicates with the outside world, whether that communication is with machine hardware, another program, or a human user. Interfaces are admittedly a fundamental part of program design, because they affect the ease, efficiency, and accuracy with which the program is used. Nevertheless, interfaces exist at a very high level of abstraction above program code. In fact, the interfaces are simply a part of what the program does—the program executes in such a way that it presents its interface to the outside world. Under traditional copyright abstractions analysis, the interfaces should lie outside the protective scope of the copyright in the program itself. This result follows naturally if the above suggestions concerning the scope of protection in programs are accepted.

Given that the copyright in a program does not protect its interfaces, the question arises whether interfaces are or can be protected at all under current intellectual property law. It is very difficult to conjure up an argument that nonuser interfaces would be copyright protected, because the interface is simply an abstract concept that does not fit into any of the traditional categories of copyright-protected work. Essentially, the interface is the method by which a program is used. As such, it is potentially eligible for patent protection, like the early typewriter keyboard arrangements. That would mean that interfaces consisting of

---

27. Copyright protection in Japan does not extend to programming languages, rules, or algorithms used to create program works. Chosakuken Hō (Copyright Law), Law No. 48 of 1970, Art. 10(3). For a detailed analysis, see Dennis S. Karjala, *Copyright Protection of Computer Software in the United States and Japan: Part II*, 13 EUR. INTELL. PROP. REV. 231, 231-34 (1991). Although Japanese judicial interpretations remain sparse, the few case authorities that exist indicate that Japan will not go down the road toward broad copyright protection of so-called "SSO" introduced into United States law by Whelan, Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222 (3rd Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987). *See* Judgment of June 20, 1989 (System Science Corp. v. Japan Technato, Inc.), Tōkyō Kōsai [Tokyo High Court], 1989, at 502 (Japan) (stating that "processing flow" is excluded from protection as an algorithm). For an analysis of this decision, see Dennis S. Karjala, *Japanese Courts Interpret the "Algorithm" Limitation on the Copyright Protection of Computer Programs*, 31 JURIMETRICS J. 233 (1991); also published 12 EUR. INTELL. PROP. REV. 235 (1990).

28. *See generally* Dennis S. Karjala, Interfaces (Nov. 7-8, 1989) (portion of proceedings, at page 269, of the 2nd International Symposium on Legal Protection of Computer Software, Software Information Center (SOFTIC), Tokyo, on file with Dennis S. Karjala).

straightforward elements composed in a more-or-less standard arrangement would not be protected outside of trade secret law. They would be protected only so long as it takes competitors to reverse engineer the program to learn the secrets of its interface.

Denying copyright protection for nonuser interfaces and affording patent protection only for nonobvious advances in interface design comports with traditional intellectual property law. Nonuser interfaces are wholly functional: they are the means by which computer programs are used. Those who would argue for a monopoly of seventy-five to one hundred years on such methods of access to the use of computer programs have the burden of demonstrating a policy basis for deviating from the intellectual property division of labor that has served tolerably well in the past. That policy basis must involve more than mere incantation that a program is a "literary work" under copyright.

2. *User Interfaces*

Still assuming that the program copyright does not protect its interfaces, aspects of user interfaces can nevertheless fall within the realm of copyright. Screen displays, for example, can be considered audiovisual works, graphic works, or perhaps compilations of textual items. Determining which aspects of user interfaces are copyright protected and which are not will again require interpretation of the statute, absent specific language aimed at user interfaces. The fundamental issue is whether functionality at the user level—user lock-in and standardization—is or should be copyright protected.[29]

Purely fanciful (nonfunctional) aspects of a user interface can be copyright protected without doing violence either to underlying policy or traditional intellectual property principles. The fanciful aspects of video game displays, which often consti-

29. It may be helpful in this context to consider the ordinary typewriter keyboard. Does the keyboard arrangement simply inform the typist which keys to punch, or does it become, with practice, a part of the instinctive way a human actually uses the machine? Omitting questions of efficiency, the arrangement of the keys is largely arbitrary. Yet, if a single manufacturer managed somehow to garner the lion's share of the initial market, users who became used to that arrangement would be reluctant to change to a different keyboard. They would be even more reluctant to switch if the first keyboard were efficient for ease and speed of typing, because competitors could offer no inducement to compensate for the difficulties involved in learning a new keyboard. A copyright in the keyboard arrangement—the predigital user interface—would then result in a very long quasi-monopoly in the first popularly used keyboard, resulting in a monopoly on the machines themselves. *See New Protectionism*, *supra* note 2, at 45-46. Although some keyboard arrangements were patented, traditional copyright would have denied protection on functionality grounds, and the absence of any reports concerning the assertion of copyright claims in typewriter keyboard arrangements suggests that there was general understanding and acceptance of this result.

tute the most significant part of the user interface, are a clear example. If Donald Duck from a comic strip can be copyright protected, so can Mario. Thus, the real issue is the extent to which functional aspects, if any, of user interfaces should be protected.

Arbitrariness in the initial choice of effecting a particular program operation through the user interface, or the existence of a wide variety of choices for doing so, should not automatically lead to the conclusion that copyright is an appropriate mode of protection. User interfaces, like all useful articles, improve via incremental advances. Some aspects of a given interface, such as that for Lotus 1-2-3, may be quite efficient. Competitors who write independent spreadsheet programs should be permitted to adopt (nonpatented) aspects that work well and to improve other aspects. The "substantial similarity" test for copyright infringement may not permit this type of incremental improvement, because an improved technological product will often be substantially similar to what it is an improvement upon.

Moreover, even an initially arbitrary choice like a particular typewriter keyboard arrangement[30] can become a standard or partial standard simply because a particular program was the first to become widely used. We should be reluctant to grant seventy-five to one hundred year monopolies or partial monopolies for merely being first, without having shown that the contribution meets the conditions for a fifteen to twenty year patent.

As noted above, if the scope of protection in the program itself is limited, through idea/expression analysis, to literal code and electronic translations (direct or indirect) of code, the protection of the user interface is relegated to traditional copyright principles. United States law is well armed to eliminate functional aspects of user interfaces from copyright protection under the tradition of *Baker v. Selden*,[31] although the two most recent and articulate authorities on the question are in hopeless conflict.[32] Developing countries might, therefore, consider express

30. *See supra* note 29.

31. Baker v. Selden, 101 U.S. 99 (1880).

32. *Compare* Apple Computer, Inc. v. Microsoft Corp., 799 F.Supp. 1006 (N.D. Cal. 1992), *aff'd* No. 93-16867, 1994 U.S. App. LEXIS 25646 (9th Cir. Sept. 19, 1994) (treating the user interface as an independent work and denying protection to functional features) *with* Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F.Supp. 203, 222 (D.Mass. 1992) (treating the user interface, including keystroke sequences, as a "nonliteral aspect" of the copyright-protected program and essentially disregarding functionality as a limit on protection); *see also* Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335 (5th Cir. 1994) (following the district court decision in *Lotus* and recognizing at least some protection for interface functionality). The *Lotus* decision is on appeal to the First Circuit Court of Appeals. For an analysis of how functionality and the doctrine of *Baker v. Selden* should be applied in that

language in their statutes defining functionality[33] and appropriately limiting the copyright protection of user interfaces to non-functional aspects.

C. Reverse Analysis of Programs

The copyright in a computer program does not, even under the most protectionist view, cover everything in the program. In particular, ideas contained in the program are not protected nor, under United States law, are systems, processes, concepts, principles, or methods of operation.[34] Yet when a program is available only in magnetically or electronically encoded object code, these theoretically unprotected elements of programs are not readily extractable, because no human being can examine and understand a complex program in this form.[35] Knowledge of these unprotected elements is often a vital aspect of creating interoperable programs and systems. Moreover, a fundamental part of the copyright balance, especially in view of the long term of protection, is that ideas and other unprotected elements in copyright-protected works should be free for all to use and build upon. All other forms of literary work exist in human-intelligible form, and anyone is free to take the unprotected elements from a publicly distributed work without the permission of the author. For this reason, reproduction of the work for the purpose of extracting unprotected elements was never an issue for traditional literary works.

Yet, to extract such elements from programs in object code form almost invariably requires making interim copies of the program through the process of decompilation of object code into a version of source code. If such interim copying constitutes infringement, copyright ends up de facto protecting theoretically copyright-unprotected elements, such as ideas and interface information necessary for interoperability. In 1989, a group of ten United States copyright law professors unanimously agreed that

---

case to deny copyright protection to the user interface of a spreadsheet program, see Brief *Amicus Curiae* of Professor Dennis S. Karjala and Professor Peter S. Menell, Lotus Devel. Corp. v. Borland Int'l, Inc., No. 93-2214 (1st Cir. 1993).

33. *See supra* note 18.

34. 17 U.S.C.A. § 102(b) (West Supp. 1994).

35. Even if the electronic representation is written out in 0's and 1's—which in any event would of course constitute a "copy" of the program—it is extremely difficult if not impossible even for a skilled programmer to make sense of it. *See Amicus Brief, supra* note 8. For an excellent description of the entire reverse engineering process for programs, and the difficulties involved in effecting reverse engineering, see Andrew Johnson-Laird, *Technical Demonstration of "Decompilation"*, 16 Computer L. Rep. at 469 (1992). *See also* Andrew Johnson-Laird, *Reverse Engineering of Software: Separating Legal Mythology from Actual Technology*, 5 Software L.J. 331 (1992).

making a small number of copies for the purpose of studying a program for possible use of its unprotected elements should be considered a fair use and therefore not an infringement.[36] This view has now been confirmed in the United States by two important courts.[37]

This problem is more easily handled under United States law because of the general fair use provision of the United States Copyright Act.[38] No general fair use provision exists in the copyright statutes of most other countries, including Japan and Germany. I have suggested that Japanese judges might consider the possibility that a technical copy for the purpose of study, as long as it does not result in a machine-readable version that can be used in a computer, should not be considered an illegal reproduction under the copyright statute.[39] A similar suggestion had been made for Germany.[40]

In Japan, it remains possible to employ an interpretation that an interim copy, for the sole purpose of study and extraction of unprotected elements, is not a reproduction that infringes the copyright. Unfortunately, the European Software Directive prohibits all forms of reverse analysis except those taken for the purpose of creating interoperable programs and makes use of the term "reproduction" with explicit reference to the rightholder's exclusive reproduction right.[41] It no longer seems possible for Germany, or other members of the Common Market, to make use of even this technical interpretation aimed at preventing the long period of copyright protection from attaching to ideas or other unprotected information contained in programs that are not related to interoperability. It might even be illegal now in

36. *LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 24-25 (1989).

37. Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992), *amended by* Order and Amended Opinion, D.C. No. CV-91-3871-BAC, Jan. 6, 1993; Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832 (D.C. Cir. 1992).

38. 17 U.S.C.A. § 107 (West Supp. 1994).

39. Dennis S. Karjala, *The First Case on Protection of Operating Systems and Reverse Engineering of Programs in Japan*, 10 EUR. INTELL. PROP. REV. 172 (1988). Professor Nakayama of the University of Tokyo has suggested that courts in Japan can adopt an implied fair use principle to solve the reverse analysis problem. NOBUHIRO NAKAYAMA, SOFUTOUEA NO HOTEKI HOGO (THE LEGAL PROTECTION OF SOFTWARE) 130-31 (1988). *See* Karjala, *supra* note 10, at 235.

40. M. Lehmann & T. Dreier, *The Legal Protection of Computer Programs: Certain Aspects of the Proposal for an (EC) Council Directive*, 6 COMPUTER L. & PRAC. 92, 95 (1990) (arguing that the term "reproduction" should be interpreted by looking at the aim of the reproduction right to assure the author of adequate participation in the economic exploitation of his work).

41. Directive 91/250/EEC, 1991 O.J. (L 122) 42, arts. 6(1) & 4(a).

Europe to reverse-analyze a program for the purpose of determining whether it is an infringing copy![42]

It seems clear that developing countries will not want to adopt software protection legislation that inhibits its own software creators from producing programs that interoperate with software and hardware from the developed countries. At a minimum, they want to leave open what remains possible under the European Software Directive. Countries that hope to become members of the Common Market may feel compelled to follow the Directive closely in order to further their economic development goals.

However, countries that feel free to take an independent course will probably want to reject the approach of the Directive. Even if reverse analysis for the purpose of achieving interoperability is the most important aspect of the problem, the Directive explicitly makes it unlawful to reverse-engineer a program for the purpose of understanding its technological ideas, its functionality, or its organizational principles, except to the extent these are related to interoperability. Both teachers and students of computer science in developing countries will almost surely want to reverse-analyze programs for the purpose of learning how they work and improving their own techniques. Probably few of them will be caught when they engage in this activity, and probably even fewer will find themselves the object of a copyright infringement action if they are. Nevertheless, it breeds disrespect for law to tell them that they are lawbreakers when they do. Reading publicly distributed copyright-protected works for the purpose of extracting ideas and other unprotected elements has always been allowed under copyright. The technological packaging of this new type of work in human unreadable form should therefore not be allowed to effect such a basic change in the traditional balance between protection and free use.

Consequently, a fair use provision applicable to computer programs that allows reverse-analysis for a legitimate purpose (such as studying the program, extracting unprotected elements, or determining whether it infringes the copyright in another program) would be a useful feature. Otherwise, those countries must rely on the courts to imply a fair use principle to resolve the reverse analysis problem, a legal process that is neither timely nor reliable.

---

42. T. Dreier, *Verletzung urheberrechtlich geschützter Software nach der Umsetzung der EG-Richtlinie*, Speech at the Annual Meeting of the Deutsche Vereinigung für Gewerblichen Rechtsschutz und Urheberrecht e.V. (GRUR), Düsseldorf (June 10, 1993).

CONCLUSION

Protection of computer program code from misappropriation made possible by modern digital methods of electronic copying and translating is a legitimate goal of the intellectual property scheme in both developed and developing countries. Computer programs are, however, fundamentally works of technology and in application bear no resemblance to traditional works of literature. Copyright law can be effective in achieving the antipiracy goal of protecting code (1) if the scope of protection is appropriately limited to that goal; (2) if interface protection is separated from any copyright in the program generating the interface; and (3) if there exists a fair use principle allowing interim copying of programs for a legitimate purpose, such as extracting and using copyright-unprotected ideas and interface information. Copyright protection that goes beyond program code to structural elements of programs or their interfaces, however, represents a much sharper intrusion of copyright into the traditional realm of patent to protect works of technology, and it requires a social policy justification that, heretofore, is wholly lacking in the software protection debate. These limitations on the copyright protection of computer software are valid worldwide but are particularly relevant to the developing countries, where a stronger scheme of copyright protection for programs would almost surely impede technology transfer and development.