COPYRIGHT, COMPUTER SOFTWARE, AND THE NEW PROTECTIONISM*

Dennis S. Karjala**

ABSTRACT

This article begins by exploring the nature of software as an area of technology similar to others except for the ease of its copying and distribution. Next, the traditional copyright protection available for technology is reviewed. Such traditional protection is either nonexistent or severely limited in scope, for fundamental reasons of social policy. The policy justification for affording software an increased level of protection over traditional technologies is piracy prevention, and increased software protection should be no greater than necessary to satisfy the policy goal.

The article then analyzes the software cases and commentary in light of the anti-piracy policy goal. The trend in the cases and much of the commentary is to protect software well beyond the level necessary to satisfy that goal, without offering a policy basis for extending protection so far. The nature of software requires that statutory interpretation and judicial citation be effected with careful reference to policy goals, as superficial application of traditional copyright analysis leads to results conflicting with optimal social policy.

Table of Contents

I. Introduction
II. Computer Programs as Technology
III. Traditional Copyright and the Protection of Technology
   A. Copyright and Technological Standardization
   B. Monopoly Through User Preference or Resistance to Change
   C. Literary Works and Technology
   D. Copyright, Compilations, and Computer Programs
   E. Proof of Infringement and Computer Programs
   F. Summary and Two Examples

---

*Copyright © 1987 by Dennis S. Karjala. All Rights Reserved.
**Professor of Law, Arizona State University. B.S.E. 1961, Princeton University; M.S. 1963, Ph.D. 1965, University of Illinois (Urbana); J.D. 1972, University of California (Berkeley).
IV. Copyright and the Protection of Computer Software
   A. Technological Standardization
      1. Copying to Achieve Operating System Compatibility
      2. Microcode
   B. User Interface Compatibility
   C. The Scope of Copyright Protection in a Program
      1. The Idea/Expression Distinction
         a. Introduction
         b. The Idea/Expression Distinction and Subsequent Authors
         c. The Black Box Approach
         d. Structure and Organization
            1. The Cases
            2. The Structural Test
            3. Commercial Competition
            4. Levels of Abstraction
         e. Summary
      2. Proof of Infringement
         a. The Direct Copying Approach
         b. The Iterative Approach
         c. A Suggested Resolution
      3. Relationship of Scope of Protection to Statutory Language
         a. Programs as Derivative Works
         b. Copying Programs in Human-Readable Form
   V. The Influence of the New Protectionism on Traditional Copyright Law
   VI. Conclusion

I. INTRODUCTION

Since the videogame cases of the early 1980s, the number of judicial decisions interpreting and defining intellectual property rights in computer software has proliferated rapidly. These decisions have, in turn, generated voluminous commentary, but the problem of protecting computer software remains the subject of intense debate. Copyright has been the primary statutory weapon, but traditional copyright itself varies widely in application to different types of works and has developed into a loose federation of legal fiefdoms unified, if at all, by a common but vague terminology and a few equally vague general principles. There is not even general agreement on the fundamental policy basis for copyright, as courts and commentators create their own combinations of incentive for investment and creation, promotion of the free exchange of ideas, and straightforward anti-misappropriation. Computer software adds yet another dimension to the debate, because reasoning by analogy permits legitimate legal arguments for placing software in more than one of the traditional copyright categories, even after nontechnically trained lawyers and
judges have mastered the essential elements of the technology. The usual technical and policy arguments within categories greatly broaden in scope because of disagreement over which category is most relevant.

Given these complications of applying copyright to computer software, it is not surprising to find a wide range of views among the commentators, both on specific applications and on general principles. In fact, it is the degree of uniformity in the judicial trend that surprises. While the courts show a variety of analytical styles, the cases indicate a widespread and highly protectionist attitude toward intellectual creations involving computer software. One may speculate that judges view the software industry as the last bastion of American technological superiority, as they read almost daily about our deteriorating ability to compete in the international markets. Software can be copied easily, and as the Japanese increase their economic domination of computer technology generally, one can understand an initial fear that Japan will swiftly take over our software markets as well unless American creativity in software is strongly protected.

In any event, and whatever the reason, the protectionist trend in the American decisions seems clear, whether it is based on an implicit view of international trade policy, an explicit view of copyright policy, or a particular conception of computer programs as copyrightable works. This article critically analyzes the judicial decisions and some of the commentary and argues that the courts have in many cases gone too far in protecting computer software, both in their notions of the scope of protection in a program and in the evidence they have accepted in finding infringement. Of course, the problem of software protection involves fundamental issues of policy, as well as unresolved empirical issues concerning what might be termed the "sociology" of computer technology—how computer programmers work and learn, how programs are created, how advances are made, and how incentives and rewards are balanced in this area of technology. Reasonable and well-informed people will differ on what degree of protection is socially optimal. Still, we must recognize that the technical aspects of copyright protection, most notably the long period of protection and the absence of any general compulsory licensing provision, have come to computer software as a single unit. If we are to expand copyright protection for programs beyond what has traditionally been available for other technologies, we must at least ask whether these particular aspects of traditional copyright continue to be appropriate. Moreover, we must ask what effect an expansive approach to program protection will have on the protection of more traditional works.

This article develops in Part II a view of the sociology of software as technology—technology that is not essentially different from other technologies in the ways it is created or used, except for the ease with which it can be copied and redistributed in competition with the original. This notion, that the availability of simple copying makes innovative software peculiarly vulnerable to the improper loss of commercial advantages over competitors, is critical.
Software and traditional technologies share a vulnerability to reverse engineering, from which traditional copyright law provides no protection, but only software can simply be copied and sold in competition without the time lags or production costs associated with other types of utilitarian products. This article refers to such simple copying as "piracy." Part III of the article reviews the traditional level of copyright protection available for technology and shows that it is either nonexistent or severely limited in scope, for fundamental reasons of social policy. The combined conclusion of these two parts is that the policy justification for affording software an increased level of protection compared to traditional technologies is piracy prevention and that the increased protection should be no greater than necessary to satisfy the policy goal.

Part IV then undertakes a detailed analysis of the software cases and commentary in light of the policy goals developed in the preceding two parts. It shows the trend in the cases, and much of the commentary, to protect software well beyond levels necessary to satisfy the antipiracy goal and without supplying another policy basis for treating software technology so differently from other technologies. It shows that the fundamentally different nature of computer programs from other types of copyrightable works requires that statutory interpretation and citation to judicial authority be effected with a watchful eye on policy goals, because superficial application of traditional copyright analysis leads to results that conflict with, rather than implement, optimal social policy. Part V then briefly analyzes two recent cases involving traditional copyrightable works that appear to have been infected by the protectionist trend.

II. COMPUTER PROGRAMS AS TECHNOLOGY

Computer programs are the technology for using computers.\(^1\) Micropograms, or microcode, convert hardwired digital logic circuitry into a general purpose computer with a specific instruction set that can be executed by that computer—the most basic instructional level in the machine. Programs at this

\(^1\)The literature concerning the legal protection of computer software is already voluminous. Much of the commentary and even many of the judicial decisions include introductory discussions outlining and explaining the technology, usually with accuracy and often with completeness. To avoid repeating much of this now well-known material, including the detailed factual situations involved in the cases, this article assumes a general knowledge of both the cases and the commentary. For general discussions of the technology in the legal literature see, e.g., P. B. MAGGS & J. A. SPROWL, COMPUTER APPLICATIONS AND THE LAW 1-19 (1987); Davidson, Protecting Computer Software: A Comprehensive Analysis, 23 JURIMETRICS J. 337, 340-48 (1983); Lynch, Steinberg, & Clapes, Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs, 34 UCLA L. REV. __ (1987) (forthcoming); Samuelson, CON'TU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form, 1984 DUKE L.J. 663, 672-90. For an excellent introduction to the actual flow of electrical signals representing information and instructions in a computer, see Holton, The Large-Scale Integration of Microelectic Circuits, 237 SCI. AM. 82 (Sept. 1977). This last source is a bit difficult for the non technically trained but can be understood with effort.

36 JURIMETRICS JOURNAL
level are said to be written in object code, or machine language. Eventually, every program that is to be run on the machine must be transformed into object code for the program to be executed on the machine, and in principle programs can be written directly in object code. In practice, however, writing in machine language is so tedious that it is rarely done. Rather, various intermediate levels of operating system programs take care of the more mundane programming tasks and put the machine in a mode that permits it to accept other programs written at a higher level of abstraction. The operating system programs thus define a higher-level, more abstract language in which application programs are normally written. Programs written in these higher-level languages are called source code. The application programs combine with the operating system to convert the general purpose computer into a special purpose machine like a word processor, spread sheet analyzer, or videogame.

When a program is stored in executable form in a computer, it is a purely physical device. Just as a key adjusts the physical environment of the lock's internal mechanism to cause it to open, so the stored executable program adjusts the physical environment inside the computer to open and close switches in a precisely timed sequence to cause the machine to play Pacman. The stored program can be translated into a form bearing a one-to-one correspondence with human-understandable symbols like printed ones and zeros; however, the machine can no more understand the written symbols 1 and 0 than it can understand the word "multiply." Still, it is precisely the one-to-one correspondence between human symbolic representation and the physical environment inside a computer that makes computers so useful. Computer programmers are in fact circuit designers, but they design with sequentially implemented symbolic instructions. The machine converts these symbols, either directly from the symbolic ones and zeros if the program is written in object code or indirectly through the operating system if the program is written in a higher-level language, into appropriate physical signals that govern the machine's operation during execution of the program.

Consequently, programs are created using symbolic representations that are understandable, at most, to human beings. They are certainly not understandable by machines, not even computers. This use of symbolic language to create programs leads naturally, almost inexorably, to their characterization as

---

2For some purposes it may be necessary to distinguish between object code written in human-comprehensible symbolic representation, usually as printed or displayed ones and zeros, and object code as physically embedded on a disk or other memory device in two distinguishable states, such as high and low voltage. Humans can read only the former, and machines can respond only to the latter. Nevertheless, the term "object code" is usually used to refer to either form.

3The often used analogy of computer programs to food recipes is inapt. It is true that light reflecting off the recipe causes neural impulses in the human brain, resulting in understanding of the instruction, "Add 1 cup milk," but the recipe does not cause the milk to be added and the cook remains free to add 2 cups of milk, or ½ cup or none at all, as he or she chooses. A computer, on the other hand, has no choice concerning the execution of instructions in a program, any more than water has a choice between flowing uphill or downhill.
“literary works” under the Copyright Act. It should be borne in mind, however, that programs resemble literary works only in their formal representation. Because programs are designed and used for causing computers to perform specific tasks, they differ fundamentally from literary works aimed at a human audience. For this reason, argument by analogy to traditional literary works is unproductive at best. In fact, it is misleading.

Notwithstanding the representational similarity of programs to literary works, programs remain the technology for using computers. They are not designed to communicate information, thought, or feeling to human beings, nor are they designed to communicate with, as opposed to physically control, computers. Even the process of program creation is more similar to the creation of other technological products than it is to the creation of books or plays, in that software engineers follow a logical and analytical process of defining the problem (functional specification), specifying the design (how to accomplish those functions), implementing the design specifications (coding), testing the prototype (debugging), documenting the procedures and specifications (to assist in later repair or improvement), and creating instructions for the user. The source code program that results from this process can of course be read and understood by others who are skilled in program technology, and to the extent that a program is designed with a view to later repair or improvement, it is intended to be so read even at the time of creation. Moreover, other programmers may be interested in reading the program for the purpose of learning from and about it. Still, human reading of a program is a distinctly secondary purpose, deeply subordinate to the primary purpose of causing the computer to perform the intended task. Thus, although the result of the programming process is expressed in a symbolic representation characteristic of literary works, the program is in fact a technological and not a literary product.

Programs are no different from other technological products in another important respect: they involve technological efficiencies. Programs are de-

---

4. "Literary works” are works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia. . . .” 17 U.S.C.A. § 101. Computer programs are universally regarded as falling within this category. E.g., Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983), cert. denied 464 U.S. 1033 (1984); NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 16 (1979) [hereinafter CONTU Report].


7. Davidson, supra note 1, at 342.

8. Plains Cotton Cooperative Ass’n v. Goodpasture Computer Service, Inc., 807 F.2d 1256, 1258 (5th Cir. 1987); Davidson, supra note 1, at 342.

9. Depending on their resolution of the basic policy issues in the software protection debate, some may refer to reading for this purpose as “stealing” rather than “learning.” At this stage of the analysis, the term “learning” can be interpreted broadly to include any taking from the original program that involves human thought in the taking process.

JURIMETRICS JOURNAL
signed to perform particular functions, and efficiencies such as higher speed or more optimal use of available memory objectively distinguish one program as better than another. Even the court in Whelan, Inc. v. Jaslow Dental Laboratory, Inc., 10 one of the more protectionist judicial decisions, recognized that efficiency is a prime concern in programs, making such structural aspects as arrangement of modules and subroutines a critical factor for any programmer. 11 Many commentators, also, have pointed out the importance of efficiency in programs. 12

Closely connected with efficiency is another characteristic common to the development of software and other technologies: innovation occurs through sequential and cumulative improvements. 13 These improvements occur not simply through the efforts of an original creator but by others who work in the same technological area and have access to the work. A civil engineer assigned to build a bridge at a particular place does not start from scratch but uses the accumulated knowledge of thousands of bridge builders who came before her, adapting it to the particular circumstances she faces. In the course of building her particular bridge, she may also make some creative improvements in design or methodology. To the extent these improvements are not protected by patent law and are visible to other civil engineers who inspect her work, they will be incorporated into later bridges. While nonpatentable improvements are by definition "obvious" in a legal sense and theoretically could be thought up independently by anyone skilled in the art, in practice engineers do look at the work of others.

There is no reason to think that software engineers are any different in the way they create their products. 14 The courts are increasingly recognizing, based on the evidence before them, that reference to existing programs in the creation of new ones is common in the industry. 15 In one case the witnesses for

---

10 797 F.2d 1222 (3rd Cir. 1986), cert. denied, 93 L.Ed.2d 831 (1987).
11 Id. at 1230. The court also noted that efficiencies and conveniences are involved in data organization, after the program structure has been determined, which make the program more or less desirable. Id.
12 E.g., Drysdale, Thoughts on Copyright: Computer Programs, 3 COMPUTER L. & PRACTICE 91, 92 (1987); Grogan & Kump, supra note 5, at 18; Nimmer & Krauthaus, supra note 6, at 41, 43.
14 See id; see also Davidson, supra note 1, at 342. Even the Whelan court seemed to recognize this point, although missing its implications: "We are not convinced that progress in computer technology or technique is qualitatively different from progress in other areas of science or the arts." 797 F.2d at 1238. See infra text accompanying note 160.
15 E.g., Q-Co Industries v. Hoffman, 625 F. Supp. 608, 612 (S.D.N.Y. 1985) (defendant programmer collected books and programs he thought would be of assistance in developing the program he wished to write); E. F. Johnson Co. v. Uniden Corp. of America, 623 F. Supp. 1485, 1498 n.11 (D. Minn. 1985)(quoting with agreement Note, supra note 13, that the "computer soft-
both sides admitted that dumping and analyzing competitors’ codes was standard industry practice.\textsuperscript{16}

In the case of bridge construction technology, and all other forms of technological development prior to the advent of computer software, incremental improvements made by subsequent engineers are protected by law only if they involve nonobvious advances sufficient to make them patentable or, in the case of trade secrets, if the new technology cannot be discerned from inspection of the product. Absent patent protection, the monopoly in creative new technology lasts only as long as it takes competitors to reverse engineer the product and tool up for production.\textsuperscript{17} The advantages of even this limited amount of lead time, however, seem to have been sufficient to provide an incentive for technological development. In fact, the absence of stronger or longer-lasting protection leads to the development of a competitive market in technology that stimulates further technological growth.\textsuperscript{18}

Given these similarities to more traditional technologies, the question is whether computer software is different in any way that calls for a different type or level of protection from that traditionally afforded to nonpatented technology. The answer is clear, well-known, and need not be belabored here. Because of the nature of programs, they can be duplicated without incurring any of the original creator’s development costs and indeed without even knowing anything about the code they contain. One can use the same tools (e.g., a computer) and the same raw materials (e.g., floppy disks) to duplicate programs of any kind, from videogames to word processing systems. Thus, the program pirate gets into competition almost immediately upon buying one copy of a commercially successful program, and he can sell at a much lower price because of the absence of development costs. Moreover, the lead time available to the innovator of tangible products, which gives not only a short-term monopoly but also a chance to develop goodwill that may carry over to help sell his product even after competition develops, is reduced to almost nothing.\textsuperscript{19}

Ease of copying therefore provides the policy basis for distinguishing computer programs from more traditional technologies. There is universal agreement that legal protection against literal copying of programs is justified, and agreement is nearly as universal that a statute styled after copyright—one that gives protection upon creation and with a minimum of formalities and cost—is the most appropriate way to effect such protection. The debate is over how much protection, if any, should be afforded beyond prohibitions against literal copying.

\textsuperscript{16}E. F. Johnson Co., 623 F. Supp. at 1501 n.17.
\textsuperscript{17}Nimmer & Krauthaus, supra note 6, at 17.
\textsuperscript{18}Id. at 22.
\textsuperscript{19}Id.
In addressing this question, we must remember that we are dealing with the protection of technology. We have decided to apply traditional copyright law directly to protect software, rather than adopt a *sui generis* copyright-style statute, even though traditional copyright was designed to protect works of a very different, primarily non-technological character. This means we must look carefully at the extent to which copyright law has been used in the past to protect technology. Expanded protection beyond traditional practice should be based on clearly articulated policy goals and be carefully tailored to meet those goals. To fail to do so runs the risk of upsetting the vague but steady balance between incentive and the free flow of ideas that has characterized traditional copyright law, should expansive software protection come to be applied to more traditional works for which different policy goals are appropriate.\(^{20}\) Moreover, any decision to expand copyright protection for computer technology beyond that available for traditional technologies must consider whether the long period of copyright protection and the absence of compulsory licensing under copyright law remain appropriate under a more protective regime.

In summary, this article accepts the following propositions as bases for analysis:

1. Technology, including software technology, grows and improves by sequential additions to the existing technological base.
2. Programs involve technological efficiencies.
3. Programmers look at existing programs in creating new programs.
4. Programs resemble literary works only in form; in the substance of their creation and use they are technology—the technology for using computers.
5. The policy basis for protecting programs differently from other creative technological products is prevention of piracy, i.e., copying not involving the time, effort, skill, or expense associated with reverse engineering of other technological products that substantially eliminates the commercial advantage accruing to technology developers in other areas in the form of a monopoly for the limited time it takes competitors to reverse engineer the product, which permits both monopoly profits and the development of goodwill, and in the form of substantially equal costs of production that competitors must incur in manufacturing a competing product.

III. TRADITIONAL COPYRIGHT AND THE PROTECTION OF TECHNOLOGY

Traditional copyright law generally protects technology only sporadically, and even technological works that are copyrightable are subject to a severely

\(^{20}\) Application of the software program protectionist approach to other areas has already begun to occur. See *infra* text accompanying notes 210-16.
limited scope of protection. 11 Before the days of computer software, most advances in technology were embodied in tangible objects that performed some useful function, which function was determinative of the object’s external appearance. Such objects are not subject to copyright protection at all because they are not “writings.” 12 Moreover, even useful articles whose external appearance is designed for aesthetic purposes are not protected by copyright in their mechanical or utilitarian aspects, and their aesthetic features are protected only insofar as they can be identified separately from, and are capable of existing independently of, their utilitarian aspects. 13

The “utilitarian” aspects even of literary works are either unprotected or receive only limited protection under copyright, pursuant to the vague but fundamental idea/expression distinction. The famous case of Baker v. Selden 14 states that where methods taught in a book cannot be used without employing the same or similar methods and diagrams used to illustrate the book, those methods and diagrams are part of the unprotected idea expressed by the work. 15 Moreover, copyrightable works like histories, biographies, scientific reports, and game rules must necessarily show a good bit of similarity in their overall presentations if they are to be true to their goals of fidelity to truth and accuracy of representation. Therefore, the general rule is that nearly verbatim copying is necessary before infringement is found. 16

These general rules and the cases implementing them in settings involving technological creativity have important implications for the protection of computer software. It is worthwhile discussing some of the non-computer-related cases relevant to the software protection problem in setting the stage for a more detailed development of this article’s theme that the courts are overprotecting software. Because of the dual nature of computer programs—a literary work in form and computer use technology in substance—it is important to be very

---

11 See generally Samuelson, supra note 1, at 728-37.
12 1 M. Nimmer, Nimmer on Copyright, § 2.18[F], at 2-208 [hereinafter Nimmer on Copyright].
14 101 U.S. 99, 103 (1880). The limits on protection of literary works that portray useful articles have also been carried over in the current statute. 17 U.S.C.A. § 113(b). See infra notes 73-76 and accompanying text.
15 Subsequent cases later interpreted Baker as distinguishing between noninfringing copying for use and infringing copying for explanation, a distinction that Professor Nimmer argues can no longer stand under the limited codification of Baker in § 102(b) of the current Copyright Act. 1 Nimmer on Copyright § 2.18[C]. In any case, Baker still stands at a minimum for the proposition that diagrams or forms demonstrably necessary in implementing the “art” described in the book (and almost as surely in describing that “art” in a different book) are considered part of the unprotected idea in the work.
careful in reasoning by analogy.\(^{27}\) Nevertheless, when the theme is protection of technology as opposed to traditional copyrightable works, both the usefulness and the limitations of many of the analogies can be brought into sharper focus. Among the issues relevant to software protection are standardization of interlocking technological products, widespread public acceptance of a particular, even if initially arbitrary, form or design in carrying out some task or method, the idea/expression distinction in literary works that describe techniques, methods, or theories, and the idea/expression distinction in factual compilations. It will be argued later that the courts in software cases have ignored the first two categories, which in fact are highly relevant, and have misapplied cases in the second two by overlooking crucial distinctions between programs and other works. The question of how traditional notions of proving copyright infringement should be reconsidered in light of the technological nature of software will also be considered. Finally, this part concludes with two software examples intended to highlight the policy issues.

A. Copyright and Technological Standardization

The technology of modern society involves many examples of interlocking parts whose form and design are only partially determined by function. A light socket, for example, must make use of an insulating material to separate electrically the two poles of the circuit, but the precise shape of the socket—its diameter and the shape and spacing of the threads—can be designed in essentially an infinite number of ways, all of which will cause the bulb to shine properly if its base fits the socket. Nevertheless, if such shapes were protected by copyright, the first fixture manufacturer to gain widespread public acceptance would have a long-term semi-monopoly on light bulbs, and the public would suffer the inconvenience that results from non-interchangeability as other manufacturers attempted to compete by introducing their own proprietary designs. For these reasons, copyright has never extended to such shapes.

One case clearly set out the policy issues nearly seventy years ago. \textit{Harvey Hubbell, Inc. v. General Electric Co.}\(^{28}\) involved an attempt to protect the shape of electrical plugs for household appliances. The plaintiff did not allege copyright infringement, presumably because of the rule against copyright protection for machine parts, but sought protection against the standardization efforts of the defendants who, by copying the shape of the plaintiff’s plugs, allegedly misappropriated the plaintiff’s ten-year investment of energy and money in developing customer goodwill. At the time the plaintiff’s plugs dominated the market in public use.\(^{29}\) The court rejected the plaintiff’s claim:

\(^{27}\)Davidson, \textit{supra} note 1, at 345; Drysdale, \textit{supra} note 12, at 91; Nimmer & Krauthaus, \textit{supra} note 6, at 21.

\(^{28}\)262 F. 155 (S.D.N.Y. 1919).

\(^{29}\)\textit{Id.} at 158-59.
No court has ever gone to the extent of permitting the establishment of a monopoly of proportions or measurements, in the absence of some patent protection. To do so would be practically to engross the particular business. . . . No one should have the exclusive privilege of selecting measurements, even though arbitrarily selected, and thus establish a particular spacing of the contacts to the exclusion of others. To do so would be to stifle competition.  

Similar reasoning applies to deny protection to the shapes of batteries, tape cassettes, computer floppy disks, and an endless list of products designed for use in conjunction with other products. Neither the creativity of the design nor the time and money expended in developing a market suffice to override the concern about monopoly.

The courts have applied this same rationale to deny protection in the copyright context. There are infinitely many ways in which a continuous temperature recording device, for example, can be constructed, and consequently infinitely many ways in which recording paper can be designed to work in conjunction with such a machine. In fact, even for a particular machine there will often be a variety of ways of arranging the labels and other marks and symbols that still permit use of the machine for its intended purpose. Nevertheless, copyright protection is denied to such forms and charts, notwithstanding their resemblance to traditional graphic works.  

Thus, traditional copyright law denies protection no matter how arbitrary or original the purported work, where recognition of copyright could create a de facto monopoly over nonpatented useful products of which the work is an indispensable part. This limitation on copyright protection in the technological context is relevant to the technological standardization problem in the case of computer software, but some of the decisions seem to be unaware of its applicability in that context.  

B. Monopoly through User Preference or Resistance to Change

The previous section considered the problems that can result from copyright protection of a crucial interface element necessary to operate machines or equipment. In that area, the existence of only a few cases is evidence of the strength and uniform judicial acceptance of the rule against copyright protection in machine parts.

Now we turn to a less firmly established variation of the same problem, in

---

30Id. at 160.
31Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947), cert. denied, 322 U.S. 801; Taylor Instrument Co. v. Fawley-Brost Co., 139 F.2d 98, 99-100 (7th Cir. 1943), cert. denied, 321 U.S. 785 (1944). These courts reasoned that copyright protection for the charts would effectively continue the monopoly in the machines beyond the time limit prescribed by patent law.
32See infra text accompanying notes 104-21.
which monopoly results from widespread consumer acceptance of a particular
design or pattern that serves as the interface not between machine parts but be-
tween a machine and its human user. Cases can be found that argue strongly for
denying copyright protection in these circumstances as well, and there is a solid
policy basis for such an approach. However, there are also cases that recognize
protection notwithstanding the resulting monopoly. These cases are of great
importance in the software arena, where the problem arises in the form of user
interface compatibility.

An example illustrates the basic issues. Consider a typewriter as a new,
patented invention. Omitting considerations of user efficiency, the arrange-
ment of the keys on the keyboard is largely arbitrary.\textsuperscript{33} As machines are sold
pursuant to the patent, typists all learn on the keyboard offered and become
used to it. Once they become skilled on a particular keyboard, they will be very
reluctant to switch to another machine. Yet, if the arrangement of the keys were
copyrightable, only the inventor of the original machine could offer type-
writers using that arrangement, even after the patent had expired. The effect
could closely approximate an extension of the patent for the seventy-five year
period of copyright protection. Moreover, to the extent that other typewriters
with differently arranged keyboards entered the market, users would suffer the
inconvenience of incompatibility.\textsuperscript{34} Many similar examples can easily be imag-
inged.\textsuperscript{35} Furthermore, although the problem is most poignantly presented when
the original invention is patented, so that the inventor has the period of patent
protection in which to lock users into his system, the same problem can easily
arise with nonpatented technology. All it takes is widespread public acceptance

\textsuperscript{33}It has been said that the traditional key arrangement was not designed for efficiency but in
fact consciously designed for inefficiency, because the early typewriters would jam at speeds
achievable by skilled typists using an efficient layout. If this is true, the fact that nearly everyone
continues to use the inefficient keyboard long after the technological need for it has vanished
strengthens the case for non-copyrightability of the arrangement, in that it evidences the strength of
our reluctance to change what we first get used to. The copyright monopoly is, moreover, even
stronger if the original arrangement is efficient, because then competitors could offer no induc-
ment to switch to machines adopting a different keyboard.

\textsuperscript{34}For example, a secretary might find changing jobs more difficult because the number of
other employers using typewriters on which he is skilled would be reduced.

\textsuperscript{35}Suppose, for example, the traditional deck of cards has been newly invented for the purpose
of playing five-card draw poker. Once the cards are widely sold, third parties begin dreaming up
new games in which they can be used, not just other poker games like stud, but also bridge, old
maid, solitaire, and all the rest. There are many ways in which fifty-two cards can be designed in
four suits of thirteen each besides using the traditional suits; the spades, hearts, diamonds, and
clubs and their arrangement are largely arbitrary. But if bridge players become accustomed to
them, they are unlikely to switch to decks using, say, grapes, apples, oranges, and bananas. A
copyright on the shapes and arrangement of the figures on the cards would force this result, how-
ever. Another example is the traditional $H$ pattern for automobile transmissions, discussed in Syn-
1978).
of one particular manufacturer’s product, for whatever reason. Once lock-in begins, it can become self-sustaining.36

The overall purpose of copyright is to draw an efficient balance between incentive for production of works and the dissemination of information for the promotion of culture and learning.37 The monopoly problem and the public inconvenience of nonstandardized useful products are real. Those arguing for protection notwithstanding should demonstrate with empirical evidence or plausible theory that the incentives supposed to result from protection lead to increased production of creative works sufficient to outweigh these disadvantages of protection. Copyright protection for things like the layout of a type-writer keyboard, in fact, quite plausibly reduces incentives for further technological development. To the extent that users are locked in to a particular format, a competitor must be able to offer a very significantly improved product before he can get into serious competition. No one would have an incentive to develop typewriters incorporating only small sequential improvements on existing products because of fear that few users would consider it worth their while to learn the required new keyboard for the sake of only modest advances.

Baker v. Selden and some of the cases following it support the proposition that copyright protection should not extend to those aspects of a system or method that, even though arbitrary or capable of being effected through a large number of different arrangements, create a user compatibility or ‘‘lock-in’’ problem through the natural reluctance of people to learn a new system once they have mastered one that serves their purposes. Professor Nimmer criticizes Baker for not limiting its holding to the specific facts before it, in which the defendant apparently did use a different arrangement of lines and columns for accomplishing the same accounting method that the plaintiff was trying to protect.38 To have rested on that ground, however, would have been to admit infringement by anyone adopting precisely the same forms as the plaintiff’s, and this ignores the user compatibility problem that would arise as different offices began adopting different arrangements for the same system of accounting. There is true social value in having standardized methods of use, and this seems to have been recognized by the Baker Court.39

Nevertheless, more current cases do not uniformly recognize user compatibility as a factor in the copyright balancing equation, at least when machines are not involved. College textbooks provide an example familiar to academics

---

36 For example, in the video recorder area two forms of technology are offered. Both do essentially the same thing as far as users are concerned, but the systems are mutually incompatible. Over time, one of the two has emerged clearly dominant (and ironically, the technologically inferior system, at least in the minds of some).

37 E.g., Whelan, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3rd Cir. 1986), cert. denied, 479 U.S. 1073 (1987). See also, Nimmer on Copyright §§ 2.18[C], [D].

38 Another early case strongly supporting the denial of copyright protection when user compatibility is involved is Perris v. Hexamer, 99 U.S. 674, 676 (1878), denying copyrightability of the symbols used to designate objects of interest on a map.
and illustrate the problem well. The real market for textbooks consists, of course, not of the students who actually buy the books but of the professors who adopt them for their courses. Anyone who has taught in college knows how much easier it is to continue teaching out of the same book than it is to switch to a new one, so competition becomes difficult when one particular book so successfully covers the material that it is widely adopted. If only the verbatim language of the popular text is protected, competing authors can follow the same general approach in the same order and possibly succeed in luring at least part of the market away from the leader. But if the overall structure is protected, adopting that structure will be infringement. One court held in favor of competition and user compatibility by finding no infringement of the copyright in a popular economics text by a competitor who carefully analyzed the book and adopted its structure,\(^{40}\) but the cases on psychology texts are mixed.\(^{41}\)

Another recent case failed to consider user compatibility in stating that the list of stocks comprising the Dow Jones Industrial Average (DJIA) was protected by copyright.\(^{42}\) The court reasoned that protection would not insulate the idea of using some other list of thirty companies as a market monitor. Perhaps the court was influenced by the constant reminders of the financial pundits that the DJIA is only one of many indicators and, in the opinion of some, that it perhaps is not a very good one for many purposes.\(^{43}\) Still, the decision fails to take into consideration that the DJIA is the most widely used measure of how the stock market is performing, whether or not it actually does what it is thought to do. At issue, in fact, was the use of the lists by a commodities exchange in communicating with its members concerning the trading of futures contracts based on the Dow averages. A monopoly on publishing such a popular list of stocks seems likely to generate inconvenience for market players that should be considered in determining the existence and scope of any copyright on it.\(^{44}\)

In sum, the typewriter keyboard example and others demonstrate the problems that can result from a failure to consider user compatibility in determining

---


\(^{42}\) Dow Jones & Co. v. Board of Trade, 217 U.S.P.Q. (BNA) 901, 906 (S.D.N.Y. 1982).

\(^{43}\) If the DJIA is in fact a highly efficient measure of market performance, a copyright on the list of stocks that makes it up would be subject to all of the objections to copyright protection for useful articles. It would be like a copyright on a list of factors useful in predicting earthquakes, secured by the scientist who first publishes his discovery of the factors that go into the prediction.

\(^{44}\) In fact, the court denied the preliminary injunction sought by the plaintiff partially on the ground that the defendant had a reasonable likelihood of success on a fair use defense. Dow Jones, 217 U.S.P.Q. (BNA) at 907-09. While recognition of the defense in the actual case might permit the unimpeded trading in futures contracts based on the DJIA, any other user of the list runs the risk of litigation and a drawn-out factual battle on the statutory fair use factors.
the extent of copyright protection. In general, traditional copyright law has recognized standardization of methodology in utilitarian works as an important factor in denying or limiting copyright protection. The recognition is only implicit in most of the cases, however, including the leading case of *Baker v. Selden*, and at least some cases hold on their facts that user lock-in does not justify a denial of copyright protection.

Ultimately, the question is one of policy: Are the existence of monopolies or partial monopolies and the inconvenience to users faced with a variety of incompatible methodologies more than balanced by creative incentives for the production of desirable works? The answer might differ depending on the type of work involved.\(^4\)\(^5\) In any event, the program user interface problem raises these same issues.\(^4\)\(^6\)

C. Literacy Works and Technology

As discussed above, by flatly denying copyright protection in the accounting method at issue rather than merely accepting the evidence that the defendant’s differing forms did not adopt the plaintiff’s expression, *Baker v. Selden* supports the proposition that copyright does not protect even otherwise arbitrary forms, charts, graphs, or arrangements when standardization of the methodology across users is an important factor in its practical implementation. *Baker* is more usually cited, of course, for its recognition of the idea/expression distinction, now codified in § 102(b) of the Copyright Act,\(^4\)\(^8\) which limits the scope of copyright even in works determined to be copyrightable. Copyright protects against copying, but only the copyrightable elements contained in the work, its expression, are protected. The ideas in the work, such as the accounting method in *Baker*, are in

\(^{4}\)For example, a copyright in the labels and decorative design of a game board, like that for Monopoly, might be acceptable notwithstanding user preference to stick with the well known Boardwalk and Park Place rather than shift to a game employing different names. The original names seem much like a trademark in which the original manufacturer has goodwill, and the loyalty of users to those names is little different from their loyalty to the name of the game itself. The situation is different when useful articles like typewriter keyboards are involved. Here the loyalty is not to the original manufacturer but only to the inertia in our own neural systems that makes relearning an old skill in a new way tedious and difficult.

Similarly, to the extent the greeting-card-creation program involved in Broderbund Software, Inc. v. Unison World, Inc., 231 U.S.P.Q. 700 (N.D. Cal. 1986), was intended as entertainment, especially children’s entertainment as suggested by the court, id. at 704, the argument for protection strengthens. In that case, however, the aspect of screen display most obviously attractive to children were the pictures of gifts, cakes, and so forth to be put together in making the cards, and these were not copied by the defendant. This case is discussed in detail infra, in the text accompanying notes 137-140.

\(^{4}\)\(^6\)See infra text accompanying notes 130-40.

\(^{4}\)\(^7\)See supra text accompanying notes 38-39.

\(^{4}\)\(^8\)17 U.S.C.A. § 102(b);

In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.
the public domain and may be taken freely, even by copying. This idea/expression distinction has over time come to take a variety of forms, depending on the nature of the work in question.

For traditional literary works, like novels and plays, the scope of protection is broad. "Expression" in these works extends beyond the literal language or paraphrases of the literal language to cover elements of structure and composition. For factual, utilitarian, and technical works, however, the concept of expression is viewed much more narrowly. Copyright protection, for example, does not extend to facts, even facts discovered by original research, and the rule extends to the order of factual presentation and even theories interpreting them. It is often stated that there must be nearly verbatim copying or wholesale appropriation before infringement of a copyright in works of this type will be found.

A clear policy basis exists for this difference in scope of protection: the social value of allowing someone to paraphrase a popular novel is minimal, and a level of protection that extends up the ladder of abstraction to protect elements of plot, character, and structure does not substantially restrict subsequent authors. Variety rather than sequential building is the social norm. However, advances in academic and technical fields are made incrementally, and free interchange of even relatively small modifications or additions in approach or substance is crucial to development. Protection beyond literal copying may therefore impede subsequent authorship of high social value.

In applying these concepts to computer software, we must first distinguish between the computer program itself and the output of the program. Output in the form of videogames is, at least under the policies outlined above, most properly analogized to novels and plays, whereas the output of, say, a word processing program should be considered a utilitarian work. The program it-

---

40 E.g., Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 55 (2nd Cir. 1936), cert. denied, 298 U.S. 669.
44 Nimmer & Krauthaus, supra note 6, at 36.
45 Videogames are treated as audiovisual works, Midway Mfg. Co. v. Artic Internat’l Inc., 704 F.2d 1009, 1012 (7th Cir. 1983), rather than literary works, but like novels and plays there is little compelling social need to limit protection to literal copying. Of course, as a game, traditional copyright law does limit protection by excluding the basic game ideas and movements from the realm of protected expression. E.g., Fryburger v. International Business Machines Corp., 812 F.2d 525, 536 (9th Cir. 1987).
46 By “output” I refer of course to the screen displays and key functions that the word processing program itself generates on the machine, not the written documents that the human operator creates with the use of the word processor. As a utilitarian literary work, or compilation of literary phrases, protection should not extend beyond the literal screen presentation. However,
self, as discussed above, has a dual nature that necessitates particular care in reasoning by analogy. Although a program is treated formally as a literary work, because it is written in symbolic representation neither the process of its creation nor its function bears any resemblance to traditional literary works such as novels or plays. The broad concept of expression in such works should have no application to programs. Programs are technology; they make computers work. At least insofar as analogy to traditional copyright law is concerned, programs should be treated as factual or technical works. Absent a clearly articulated policy reason for deviating from the traditional balance drawn by copyright law between the creation of incentives and promotion of technological development, infringement should be limited to verbatim or near verbatim copying of the literal language of the program, that is to say, the literal coding. We consider below how the antipiracy policy offered herein as the justification for program protection distinguishes programs from traditional factual works in determining the scope of protection.

Even when the analogy of programs to works of fact rather than works of fiction is recognized, however, it is important to note an important difference: programs involve technological efficiencies. Moreover, they do not describe how to achieve technological efficiency; rather, they are the means for achieving it. The policy basis for the rule limiting protection in factual and technical works to near verbatim copying is often stated to be the existence of only a limited number of ways of expressing the ideas contained in such works. Conversely, if the court perceives a wide variety of ways of expressing the same idea, less similarity may be required in finding infringement.

If a program is viewed as performing a general overall function, like word processing, there are probably as many different ways of writing it as there are of writing *Hamlet*. But that does not mean that structural aspects of a program sufficient to infringe an assumed copyright in *Hamlet* if the program were a play should necessarily be deemed an infringement of a copyright in a program because output of this type presents the user with a machine that is manipulated to perform work. It is also analogous to the typewriter keyboard discussed in the previous section. See supra text accompanying notes 33-37. Because standardization of such user interfaces is a major social goal, copyright in such output is properly denied altogether. See infra text accompanying notes 130-40.

58 See supra text accompanying notes 3-9.


50 See infra text accompanying notes 79-83.

54 Programs do describe means for achieving efficiency indirectly, insofar as another programmer can read the program and translate it in his mind into internal actions of the computer. But the direct purpose for writing the program is to operate the computer, not to describe anything. See supra notes 8-9 and accompanying text.

60 *E.g.*, Lansberg v. Scrabble Crossword Game Players, Inc., 736 F.2d 485 (9th Cir. 1984). In the extreme case of very simple subject matter that admits of only a few possible forms of expression, copyrightability may be denied altogether. Morrissey v. Procter & Gamble Co., 379 F.2d 675, 678-79 (1st Cir. 1967).

from which the structure was taken. It is likely that no one will ever do better than Shakespeare at the Hamlet story, and in any event there seems little social harm in making people wait seventy-five years before being allowed to try. But if someone achieves the equivalent of Hamlet for certain aspects of a word processing program, and someone else does the same for other aspects, society should not have to wait seventy-five years before it is able to use a single program incorporating both optimally efficient creations. Or, at least, in deciding on policy grounds that copyright should protect such technologically efficient products, the policy basis should explain the reason for departing from traditional copyright notions. Except perhaps until now with the advent of computer software, courts that have recognized the problem have uniformly rejected attempts to protect technological efficiency under copyright.

D. Copyright, Compilations, and Computer Programs

We must also distinguish programs from traditional “sweat of the brow” fact works, such as directories. Computer programs are not properly analogized to such works, although computer databases are. These differences must be recognized when applying judicial authority in this area.

The question is complicated by judicial confusion over the basis for protecting factual compilations. The copyright statute expressly protects compilations to the extent of the material contributed by the author, but the definition of “compilation” seems to look to the selection or arrangement of the materials or data comprising the compilation as the basis for treatment as a work of au-

---

62 Hamlet, of course, is not a factual work, but because program efficiency depends as much, or more, on structure as on coding, programs must be distinguished from either type of work when efficiency is a concern. The example of Hamlet rather than, say, a physics text, is used here because the CONTU Report includes testimony from an expert in computer programming in which this analogy to Hamlet is used to justify the copyrightability of programs. The expert conceded the possibility that there might be differences in efficiency between the various programs, and the Report itself assumes that programs contain ideas that other programmers are free to use. CONTU Report at 20 & n.106. It is difficult to imagine what those ideas might be if not program structure.

63 Of course in this simple example, presented to bring the policy issues to the fore, independent creation by other programmers would probably lessen the time required for availability of the single optimal program. Protection of the optimal structures, however, means that later versions must be independently created and that independent creation must be provable. This forces software engineers to adopt unusual procedures, such as “clean room” development, that are not based on any technological need and may actually impede long-term development of software. See infra text accompanying notes 115-17. It also ignores the reality that programmers, like all other engineers, do look at what others have done to see if and how they are doing it better. A better example might be ten or twenty word processing systems, each with features different from or better than those of at least some of the others. The policy issue is whether the creators of each system should have to do all improvements from scratch, without looking at how the others are doing things, or whether we can rely on competition among the manufacturers of these systems to provide the incentives for improvement, as we do with reverse engineering in other areas of technology. Nimmer & Krauthaus, supra note 6, at 21-22.

64 17 U.S.C.A. § 103.
authorship. As individual facts are not protected, however, this selection and arrangement rationale for copyright protection would leave works like telephone books without copyright protection at all, because neither the selection standard (all the people in a particular geographical area) nor the arrangement (alphabetical order by last name) is original. Seeing social value in providing an incentive for the often costly and time-consuming production of such factual compilations, some courts have deviated from the selection and arrangement rationale in order to supply copyright protection.

These courts have found originality in the first author's efforts in collecting and putting together the materials. The leading proponent of this approach among the commentators is Professor Denicola, who argues for the notion of "authorship in the collection" as both a descriptive and a normative theory for the law of factual compilations. Such an approach is vital for coherently grounded protection of computer databases, which in essence are no more than unintelligible jumbles of binary signals. For many databases, if the effort of collecting and placing the information into the computer is not protected, there will be nothing to protect.

Some commentators have concluded that copyright has moved away from its traditional policy of encouraging disclosure of ideas and balancing incentives against the free flow of ideas, to become a general anti-misappropriation scheme, protecting the fruits of intellectual labor. To the extent this position is based on the factual compilation cases, however, it must be noted that some recent decisions seem to deviate from the "sweat of the brow" basis for copyrightability of factual compilations and to be returning to the selection and ar-

---

65 Id. § 101:
A "compilation" is a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship.


68 Id. at 531. Of course, the program organizing the database provides the user interface necessary to sort out all of these signals, but that program is separately protected; moreover, the creator of the database may not own any proprietary rights in that program.

69 An example is the Lexis legal research system. The cases themselves are in the public domain, all cases are included (so there is no selection scheme), and the organization by the program is standard (by original reporter, subject matter area, and so forth). Yet placing that vast amount of data into the database is very costly, notwithstanding that the process is purely mechanical. See generally Karjalaa, Protection of Computer Databases under Japanese Law, 9 Eur. Intell. Prop. Rev. 267 (1986).

70 Note, Copyright Protection of Computer Program Object Code, 96 Harv. L. Rev. 1723, 1741 (1983); see also Denicola, supra note 67. For an extended judicial discussion of both the history and rationale of the misappropriation doctrine as applied to factual compilations, see National Business Lists, Inc. v. Dun & Bradstreet, 552 F. Supp. 89, 92-95 (N.D. Ill. 1982).
rangement rationale.\textsuperscript{71} Much more importantly, it is impossible to maintain that copyright now protects against the misappropriation of the fruits of \textit{all} kinds of intellectual labor. Every new technological product embodies the fruit of intellectual labor, yet only those that can qualify as "writings" have any hope at all of copyright protection. Moreover, writings embodied in a useful product are expressly denied protection under the current statute except to the extent that the design can be separated from the utilitarian aspects in the case of pictorial, graphic, or sculptural works,\textsuperscript{72} or to the extent permitted by \textit{Baker v. Selden} in the case of works that portray useful articles.\textsuperscript{73}

Consequently, even if the protection of factual compilations under copyright is grounded on notions of anti-misappropriation, there is no basis for extending that notion to computer programs. While Congress clearly intended to protect computer programs under copyright, there is no evidence that Congress intended to protect all fruits of intellectual labor embodied in programs. On the contrary, under the 1976 revisions, Congress strengthened the rule against copyright protection for utilitarian articles.\textsuperscript{74} Under the current statutory language, a computer program fits easily within the definition of "useful article,"\textsuperscript{75} and the symbolic representation of the program portrays that useful article. Therefore, all of the prior judicial interpretations limiting copyright

\textsuperscript{71} \textit{Financial Information Inc. v. Moody's Investors Service Inc.}, 1 U.S.P.Q.2d (BNA) 1279, 1281 (2nd Cir. 1986) (time-consuming but clerical collection of facts on cards lacks originality required for copyright); \textit{Cooling Systems & flexibles, Inc. v. Stuart Radiator Inc.}, 777 F.2d 485, 491 (9th Cir. 1985) (an author can claim "only an original manner of expressing ideas or an original arrangement of facts"); \textit{Rockford Map Publishers, Inc. v. Directory Services Co.}, 768 F.2d 145, 149 (7th Cir. 1985) ("[C]opyright depend[s] on the fact that the compiler made a contribution—a new arrangement or presentation of facts—and not on the amount of time the work consumed"); \textit{Gem Products, Inc. v. Robertshaw Controls Co.}, No. CV-85-7283 FFF, at 8 (C.D. Ca. Feb. 26, 1986)(copyright does not protect facts, only the arrangement and manner of expression).

\textsuperscript{72} \textit{17 U.S.C.A.} § 101.

\textsuperscript{73} \textit{Id.} § 113(b);

This title does not afford, to the owner of a copyright in a work that portrays a useful article as such, any greater or lesser rights with respect to the making, distribution, or display of the useful article so portrayed than those afforded to such works under the law . . . in effect on December 31, 1977, as held applicable and construed by a court in an action brought under this title.

\textit{See} 1 \textit{Nimmer on Copyright} § 2.18[B], at 2-197 n.15. The court in \textit{E. F. Johnson} overlooked the § 113(b) limitation on the scope of protection in useful articles when it concluded that programs, as literary works, are not subject to the similar limitation in § 101 regarding pictorial, graphic, and sculptural works. E. F. Johnson Co. v. Uniden Corp. of America, 623 F. Supp. 1485, 1498 (D. Minn. 1985).

\textsuperscript{74} Samuelson, \textit{supra} note 1, at 731-32.

\textsuperscript{75} \textit{17 U.S.C.A.} § 101;

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. An article that is normally a part of a useful article is considered a "useful article."

A program has an intrinsic utilitarian function, which is to cause a computer to operate as intended by the programmer. That function is not merely portrayal of appearance or conveyance of information, because the program is intended not simply (or even primarily) to be read, but rather to be used in a computer.

FALL 1987
protection in technology continue to apply.\textsuperscript{76} Computer programs are technology. They are the fruit of intellectual labor, but copyright law does not necessarily protect that fruit. If the scope of protection is to be expanded for programs beyond that afforded to other technologies, the expansion must be based on policy grounds and not by analogy to the "sweat of the brow" factual compilation cases. The real relevance of the compilation cases is that they show judicial willingness to depart from traditional copyright concepts, such as the idea-expression distinction, when necessary in light of the nature of the work and optimal social policy. A departure is necessary for copyright protection of program works as well, but the very different natures of programs and compilation works demands a departure of a different type.

E. Proof of Infringement and Computer Programs

Traditional copyright protects against copying of expression. Therefore, to establish the case for infringement, a plaintiff must prove both copying and that what was copied included expression. Copying can be proved by showing the defendant's access to the original work plus substantial similarity of either idea or expression sufficient to convince the trier of fact that the similarities more likely than not resulted from copying rather than independent creation, which then shifts the burden on the issue to the defendant. Proving "expression" involves the basic scope-of-protection issue and requires distinguishing the underlying ideas contained in the work from the expression of those ideas, because no infringement arises if only ideas are taken.\textsuperscript{77} This in turn involves the basic policy balance between how much protection should be afforded the original author and how much freedom should be allowed to subsequent authors.\textsuperscript{78}

The previous two sections discussed the inapplicability of the traditional literary work and compilation cases in determining the scope of protection in computer programs, and the limited applicability of the historical, technical, and fact work cases. There are some additional differences between programs and other copyrightable works, even technology-related works, that are relevant to proof of infringement, and it is worth looking at those differences briefly before moving to a more detailed analysis of the software cases themselves.

The first difference is that there often seems to be direct or at least highly convincing indirect proof of copying in the case of programs. Thus, rather than require access plus substantial similarity of the programs as a whole (or in significant part), we are often comfortable in inferring copying from only a few

\textsuperscript{76}See supra note 73.

\textsuperscript{77}Frybarger v. International Business Machines Corp., 812 F.2d 525, 529 (9th Cir. 1987); Walker v. Time Life Films, Inc., 228 U.S.P.Q. (BNA) 505, 508 (2nd Cir. 1986); Cooling Systems & Flexibles, Inc. v. Stuart Radiator, Inc., 777 F.2d 485, 491-93 (9th Cir. 1985); 3 Nimmer on COPYRIGHT § 13.03 [A].

\textsuperscript{78}Nimmer & Krauthaus, supra note 6, at 31-32.
isolated instances of "surprising" similarity. At the coding level, complex computer programs consist of thousands of lines, many of which are nonfunctional. Nonfunctional code is often used for purposes of assisting other programmers (within the company) in understanding how the program works so they can repair it when necessary and also make later improvements. Nonfunctional code is also sometimes present simply by mistake, as when a program is changed but not all of the code intended to be deleted gets removed. It is also sometimes deliberately inserted as a "trap" for copiers. Whatever the reason for its presence, such identical nonfunctional code seems so unlikely to arise by coincidence that it is strong evidence of copying, at least to that extent.

The real question in many cases, therefore, is whether the copied code included copyrightable elements of the first work, which is the basic policy issue. In resolving this policy question, we must continue to bear in mind that programs are technology and that progress in this technology is made in much the same way as other areas of technology—by sequential additions and improvements to the existing base. But here a second potential difference of programs from traditional works may call for a difference in the judicial approach to determining infringement. Under traditional copyright analysis courts usually formulate this policy issue in terms of the idea/expression distinction, but there is a real question of whether courts can efficiently make this distinction in the context of computer programs. With traditional works, the issue is deliberately left vague. We rely on the judgment and good faith of the triers of fact, usually without the assistance of expert witnesses. Computer programs, however, cannot be read like novels, even by experts, let alone by lay judges and jurors. Programs may be comprised of thousands of lines of code, and there are at least some ways in which essentially direct copying can be effected and yet produce a coded result that appears quite different in its symbolic representation. Paying experts to wade through this dense forest of symbolism and to explain their conclusions to the trier of fact can consume a vast amount of resources. A method for proving infringement that avoids these costs seems clearly desirable.

The circumstances reported in the cases thus far in fact suggest such a method, when those circumstances are conjoined with the policy goal of preventing piracy while interfering to the minimum extent possible with cumulative innovation. The method focuses on protection against true piracy. Piracy consists of copying not involving the time, effort, skill, or expense associated with reverse engineering of other technological products. For other products, that time, skill and expense of reverse engineering maintains some commercial

---

80E.g., Sid & Marty Kroft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977); Arnstein v. Porter, 154 F.2d 464, 468 (2nd Cir. 1946), cert. denied, 330 U.S. 851 (1947); Nichols v. Universal Pictures Corp., 45 F.2d 119, 123 (2nd Cir. 1930), cert. denied, 282 U.S. 902 (1931).
81Davidson, supra note 79, at 1085. The easiest example is the comparison of code that has been run through a different compiler.
advantages for the initial developer who has a monopoly for the limited time it
takes competitors to reverse engineer the product. This limited-time monopoly
permits both monopoly profits and the development of goodwill, important ad-
vantages since competitors, after reverse engineering, will still incur substan-
tially equal costs in manufacturing a competing product. To the extent soft-
ware competitors are forced to undertake development activities similar to
traditional reverse engineering, the first aspect of the piracy problem is elimi-
nated. And if such reverse engineering activity in the case of software approxi-
mates actual development costs not recovered during the original's period of
monopoly, the competitor will not be able unfairly to undercut the price
charged for the original. Moreover, if the "competitor's" program does not
compete with the original program, there is no piracy. On the contrary, the
advantages to the second programmer in adopting the fruit of another's intel-
lectual effort results in a social benefit in the form of expanded availability of
computer technology, typical of the reliance on the technological innovations
of others found in all fields of technology.

When copyright infringement in a program (as opposed, for example, to
the output of a program) is asserted, we should cease the attempt to distinguish
idea from expression. Rather, we should require that the plaintiff present credi-
ble evidence of direct copying, which will often be available—at least in true
piracy cases—in the form of mistakes or nonfunctional code in the alleged copy
identical to code in the original. An inference of direct copying may also be
permitted from a showing of comprehensive duplication of functional sequence
and structure. Once such evidence is presented, the burden shifts to the de-
fendant of proving either truly independent production (via a "clean room," for example) or that the copying did not confer any substantial commercial ad-
vantages of piracy. The former could be shown by evidence of a paper trail
demonstrating legitimate reverse engineering activity; the latter by evidence
that there was no substantial competition with the original program. Reverse
engineering could be shown by providing the source code derived from decom-
piling the original object code, and by evidence that such source code was ex-
amined carefully in creating the defendant's program. Careful examination
considers not just the individual code lines but also the structure and sequenc-
ing of operations, for the purpose of determining what parts of the original can
be improved and how improvements can be effected.

It should not be required that the defendant actually find every problem or
inefficiency in the original but only that he make an honest effort to conduct
such an examination. If he does, he will of necessity have created a paper trail
showing what he did, and it should not be difficult for him to carry his burden.
Engaging in such an effort takes both time and money, and returns to the origi-
nal creator the advantages that are otherwise lost to the pirate. Once piracy is no

87See supra text accompanying note 19.
83See Nimmer & Krauthaus, supra note 6, at 48-49.
longer a concern, the free flow of innovative technological ideas should be our primary consideration. "Honest effort" will probably be as difficult to define precisely as the idea/expression distinction currently used with literary works. In the end, we still rely on the judgment and good sense of the trier of fact, but the focus is on the efforts of the defendant in seeking to create a better product, and most particularly, on whether those efforts required sufficient investment of time and money or otherwise resulted in a product for which the headstart and/or production cost advantages were not substantially different from those available to reverse engineered products in other fields of technology. The focus thus shifts, from copyright's traditional emphasis on how closely the defendant's work product resembles the plaintiff's, to the nature and amount of the defendant's effort in producing the product. This is appropriate to technology protection, because improved technological products almost always contain many similarities to the original, but we do not bar them from the market simply for being similar. In fact, we encourage the making of such improvements.

If we are to penalize building on the efforts of others for reasons that do not relate to the piracy problem, we should be clear about the policy grounds for doing so. The opinion in SAS Institute, Inc. v. S & H Computer Systems, Inc. illustrates the very different focus that results from failing to treat programs as technology. Here the defendant targeted the plaintiff's statistical analysis program for conversion into another language so that it could be run on an incompatible computer. The court emphasized that the defendant saved substantial time and effort by "expropriating" the plaintiff's proven design, in avoiding false starts, dead ends, and other fruitless efforts. The court failed to realize that advantages of this type are available in nearly all cases of technological reverse engineering. It is precisely because we do not want to force every engineer to go through all the steps necessary to reinvent the wheel, including false starts, that we have given such limited protection to nonpatented technological innovations. In SAS Institute, the defendant's program did not compete with the plaintiff's original program. When viewed as technological development, the defendant's activity is little different from using someone else's idea for an internal combustion engine inside a four-wheel vehicle to design a motorcycle.  

---

85 The court also analogized the defendant's activity to the theft of the design for a skyscraper, thus avoiding all of the design costs. Id. It did not even notice that while stealing the documents would be punishable under the laws of theft, building the skyscraper from the design documents would most definitely not be copyright infringement.
86 In particular, the potential for the original programmer to write a second program aimed at the market the defendant has targeted for his converted program is no more deserving of protection than the automobile manufacturer's potential for entering the motorcycle market. See infra text accompanying notes 160-65 for a discussion of the social utility of the conversion of programs to run on computers with which the original program is incompatible, as well as additional arguments for not treating such activity as copyright infringement.
F. Summary and Two Examples

We begin with the proposition that programs are technology, notwithstanding their formal appearance in symbolic representation. As in all other areas of technology, software engineers look at the work of others in their effort to add to and improve on technological innovation, and they carry their skills and their memories with them as they move from job to job. Traditional copyright law does not protect technology at all to the extent it is embodied in tangible products, with the result that copying of such technology is legally restricted only by patent or trade secret law, and in the latter case only to the extent the technology is not susceptible to reverse engineering. Moreover, copyright does not protect technological ideas—means and methodologies for making machines work better or making processes more useful—embodied or described in traditional copyrightable works, such as books or articles. Absent some clearly expressed policy reason for deviating from these traditional levels of protection for technology, we should not treat computer programs any differently.

One policy basis for a different level of protection for programs is the case with which they can be pirated: it is not necessary to reverse engineer the product, or even to understand the ideas taken, in order to take them. Moreover, there is essentially no lag time between buying a single copy of the desired original program and getting into competition with the original on a large scale. Making an exact copy of a program, without analysis or any attempt to understand and improve upon it, should clearly be, and clearly is, a copyright violation. On this point, at least, there is universal agreement.

On the other hand, there is no unfair advantage, at least when viewed in the light of the level of protection afforded to other technologies, to the competitor who engages in legitimate reverse engineering of a program. True reverse engineering gives the original maker the same headstart in the market available to developers in other technological areas and provides the competitor in the usual case with little if any advantage in reduced development costs. To the extent the original product offers a new function not previously offered to the public, competitors are likely to wait even longer before incurring the cost of reverse engineering, to see whether demand is sufficient to justify the reverse engineering costs. This permits the original maker even more time to develop goodwill and customer loyalty.

Consequently, rather than attempting to determine whether there is substantial similarity of protected elements, which requires an exhaustive examination of program details and then a policy judgment concerning whether what was taken was "expression," the courts can go straight to the policy question of whether what the defendant did was an honest attempt at a socially desirable activity, namely, reverse engineering for the purpose of improvement, or else socially undesirable piracy. In the end, this may well be what courts are doing.
anyway, but it is hidden under a veil of idea/expression verbiage. The approach suggested here not only legitimizes this general approach but also gives the court a standard for judging what activity is desirable and what is not.

The facts of *NEC Corp. v. Intel Corp.* currently still in litigation, provide a good illustration of the basic policy issues involved in the copyright protection of program technology. The work in question is the microcode that defines the basic machine language instruction set of a popular microprocessor. For the present, we need not concern ourselves with the special features of microcode that may distinguish it from other computer programs in the software protection analysis. The plaintiff, one of the largest manufacturers of semiconductors in the world, set out to develop a microprocessor that would be compatible with the defendant’s but would include improvements in the form of a more powerful instruction set, faster speed of operation, and lower power consumption. Compatibility was essential, because the defendant’s chip was the heart of all IBM-PC-compatible personal computers, for which tens of thousands of applications programs were already available. The plaintiff engaged in a full-scale development effort, including significant changes in the hardware and chip architecture, and ultimately succeeded in achieving the desired improvements in a compatible chip. It was introduced into the market more than five years after the first offerings of the defendant’s chip.

Given that writing the actual microcode is a relatively small part of the cost of overall chip development, it seems almost incredible that a company in the plaintiff’s position would copy microcode after independently developing everything else, unless they thought copying was necessary for compatibility or was dictated by the hardware environment. Even assuming copying, how-

---

87 No. C-84-20799-WAI (N.D. Cal.). NEC sought a declaratory judgment that its microcode did not infringe, so the positions of plaintiff and defendant are reversed from the usual copyright case.

88 The factual basis for the example is taken from articles by attorneys for the two parties in the *Intel* case, which appear to be adaptations of trial briefs. Dunlap, *NEC v. Intel: A Challenge to the Developing Law of Copyright in the Protection of Computer Programs*, 3 COMPUTER & HIGH-TECH L.J. 3 (1987) (representing Intel); Hinckley, *NEC v. Intel: Will Hardware Be Drawn into the Black Hole of Copyright?*, 3 COMPUTER & HIGH-TECH L.J. 23 (1987) (representing NEC). The analysis in the text does not depend on the actual facts of the case, as the example itself and the facts assumed in it form the basis for analysis. If the facts in the *Intel* case are different in any relevant respect from those assumed in the example, then obviously a different analysis may be appropriate. In the actual case, the court has ruled that Intel’s microcode is copyrightable, but the issue of infringement remains to be decided. *Intel*, Partial Findings of Fact and Conclusions of Law (Sept. 22, 1986).

89 There is a respectable argument that microcode should not even be considered a computer program at all and therefore that it should not be eligible for copyright protection. On that issue, the court has already ruled in favor of copyrightability. There are also substantial issues, under the traditional idea/expression analysis, of merger of idea and expression, because as computer programs go, microcode is very elementary and highly constrained by the hardware environment in which it is used. See infra text accompanying notes 122-23.

90 For NEC’s side of the actual story, see Hinckley, supra note 88, at 25-29.
ever, to find copyright infringement is to give to this most fundamental form of computer software technology a level of copyright protection never before afforded to a technological product. Because of the absence of compulsory licensing under copyright, the defendant would not only retain an economic monopoly through the extraction of royalties on sales of the plaintiff’s chip; the defendant can wholly enjoin the sale of the plaintiff’s chip, thereby depriving the public of the advantages of a technologically superior product. Moreover, this position will be retained for seventy-five years, or at least until the original chip becomes obsolete through later technological developments. This seems a very high price for the public to pay to remedy a “wrong” that would not be at all wrong or unfair if it involved any type of technology other than one that effects machine operation through manipulation of human-understandable symbols. The plaintiff’s long-term, full-scale development effort assures that the plaintiff’s product has no competitive advantages over the defendant’s except in product quality and, conceivably, more efficient development and production methods.

In the Intel case, noninfringement probably can be found on more traditional idea/expression reasoning. Given the nature of programs in general, however, and the manner in which program technology develops, it seems a waste of resources to spend so much time battling over whether particular similarities in complex programs, or even simple programs like microcode in a complex hardware environment, are idea or expression. Deciding whether to label the contested item “idea” or “expression” is, and always has been, a policy issue, but the policy factors relating to programs are very different from those that go into separating plays from historical works. The labels “idea” and “expression” may serve as a useful mnemonic for the relevant policy choices in literary works, but in technologies those same labels lead courts astray. For software, a form of technology, the thing to be stopped is piracy, and it is very difficult to see piracy—an unfair advantage obtained through direct and non-analyzed copying and largely nonexistent development costs—on the facts in this case.

A more difficult case but one that pushes rather dramatically at the boundary between computer programs and traditional technology is Midway Mfg. Co. v. Strohon, a videogame case in which the defendant created a speeded-up version of the plaintiff’s copyrighted Pacman game. With coin-operated videogames, apparently, many players become so skillful at a particular game that they can pattern play for long periods and/or eventually lose interest. In either case, the machine ceases to be profitable. The defendant owned some machines

---

89In the actual case, apparently not even Intel asserts that all of the microcode contained in Intel’s chip was copied. Dunlap, supra note 88, at 16-17 (Intel’s expert testified to similarity in the sequencing of fifty-four of NEC’s eighty-eight microprograms).
89Injunction is one of the remedies sought by Intel in the actual case. Dunlap, supra note 88, at 4.
90See infra text accompanying notes 128-29.
and attempted to increase the return on his investment by providing a greater challenge to skilled players. He did more than merely speed up the play, however; he changed the shapes of the characters and otherwise adapted the screen display to the extent that his game did not infringe the plaintiff's copyright in Pacman.95 This is very important, because it means that the defendant's output was not a derivative work but rather an original game.

Nevertheless, in producing his new game, the defendant found he had to change only about eleven percent of the code in the program governing the game's operation.96 The rest, we may assume, was adopted verbatim from the plaintiff's program. By analogy to traditional literary works, the court reasoned that it would be infringement to take one chapter from a novel or one volume of a multivolume work. Consequently, it found that taking eighty-nine percent of the plaintiff's code infringed the copyright, and it enjoined marketing of the defendant's game.97

Programs are technology, however, and the analogy to traditional literary works is inappropriate.98 To see the difference that results from analogy to traditional technology, consider a variation on Professor Samuelson's airplane wing example.99 An airplane manufacturer studies a competitor's unprotected but sleek wing design and develops a variation that makes airplanes go even faster. Not only is there no infringement of the competitor's intellectual property rights, but we applaud the new contribution to technological development that inures to the benefit of all. We would not ask how much money the original wing developer spent on its development, nor do we call it misappropriation or piracy when the second developer adopts the design with improvements.

Under the analogy to traditional technology, the defendant in Strohan, too, has supplied a social benefit.100 He took what appears to have been an obsolete game and made modifications to create a new game for which there was a market demand. It is true that he took much of the plaintiff's code directly, probably without extensive analysis. That is the only factor, however, that makes the case even close. The policy issue is whether we want to make program technologists redesign entire programs that they use as a basis for making improvements, even when there is no need for improvements like increased effi-

95Id. at 747.
96The eighty-nine percent unchanged code figure was based on actual sequencing instructions as opposed to data that appears on the screen. Presumably, if the different data had been considered, the percentage would have been lower. The court also found that the locations in memory of these instructions were ninety-seven percent duplicated. Id. at 752-53. The court did not attempt, however, to assess the relevance of the substantial identity of instruction location.
97Id. at 753-54.
98See supra text accompanying notes 47-58.
99Samuelson, supra note 1, at 726.
100This, of course, assumes that new videogames are a social benefit. This author does not actually accept that proposition, but believes that copyright is not the proper method of restriction.
ciency. An argument in favor of such a requirement, which implies a greater level of protection for program technology than for other technologies, is that we might find a rash of piracy masquerading as "improvements," where the second developer makes only a few minor changes and claims eligibility to sell the improved version in competition with the original. Courts could easily bog down in deciding what types of improvements are sufficiently valuable to justify the taking.

On the other hand, we must recognize the costs involved in the Strohon approach. In a case like Strohon, the number of highly skilled players is smaller than the number of players in general, and the potential market may not justify full-fledged reinvention of that particular wheel. Moreover, the defendant's game did not compete with the plaintiff's, because the whole point of its development was to appeal to players who had tired of the plaintiff's game, so the antipiracy policy basis for protecting program technology under copyright is absent. If we are to protect program copyright holders in the plaintiff's position in cases like this, it may be appropriate to consider compulsory licensing schemes that permit improvements but require a royalty to the original developer. Under the Strohon approach, the improvement is simply enjoined off the market.

IV. COPYRIGHT AND THE PROTECTION OF COMPUTER SOFTWARE

The reported cases involving computer software already show a wide variety of fact patterns and a concomitant variety in the nature of the legal problems, both policy and technical, that they raise. There are problems of how copyright protection affects technological standardization—the compatibility

---

101 That is to say, efficiency in a videogame program is probably less important than it is in many other application programs, because if the screen appears to the user as intended then the purpose is served. It makes little difference that particular instructions are processed in fewer microseconds rather than four, because users cannot detect the difference.

102 See infra first full text paragraph following note 164. It does not meet this point to argue that the plaintiff may have intended to enter the speed-up market. The defendant's game was a new game, not a derivative work, and the plaintiff had no rights to that game. Moreover, in the speed-up market, defendant has no unfair advantages over plaintiff. Whether the defendant's program was a derivative work, of course, is another matter, but to follow the literal statutory language in concluding that it was only begs the policy question of whether what the defendant did should be considered infringement. In any other field of technology, changing ten percent of an existing product to create a new one would be considered useful and innovative behavior. Moreover, there is great danger of overproduction in the case of program technology when the statutory definition of derivative work is resorted to. See infra text accompanying notes 197-203.

103 Before opting for copyright protection for programs, the Japanese were seriously considering a sui generis Program Rights Law that would have included a compulsory licensing provision, although it is not clear how that statute would have dealt with the problem in Strohon. Karjala, Lessons from the Computer Software Protection Debate in Japan, 1984 Ariz. St. L.J. 53, 68-69, 73-74.
of one manufacturer’s product with another’s. There are problems of whether copyright protection should be tempered to avoid undesirable user lock-in and to promote standardization across users—so-called “user interface compatibility.” Both areas raise fundamental questions of policy. Beyond those, there is the general problem of determining the scope of protection in a program, which raises a whole set of both policy and technical questions. These questions can be formulated in different ways: as an issue under the traditional idea-expression distinction, as whether a particular program is a derivative work, as whether copying for the purpose of studying a program constitutes infringement, or as a question of how “substantial similarity” should be defined for purposes of determining infringement. Finally, because in application to computer programs copyright is for the first time protecting technology as such and not merely explanations or descriptions of technology, there is the problem of whether any expansion in protection deemed necessary for programs will be carried over to modify the traditional copyright balance in other areas. Many of these issues overlap, but this general breakdown provides a useful framework for sorting out and analyzing the various approaches that have been adopted by the courts and suggested by some of the commentators.

A. Technological Standardization

Technological standardization refers to compatibility of one manufacturer’s products with those of another. A preeminent example is the shape of the electrical plug for household appliances.\(^{104}\) Once a particular shape is in general use, manufacturers will tend to make products using only that shape, and new home builders will be more likely to use sockets accommodating that shape in order to accept the maximum number of plugs without the need for adaptors.\(^{105}\) Thus, initial market leadership in plug shape can become market dominance through the ratchet effect.

Compatibility problems of this type are arising in an increasing number of ways in connection with computer software. There is first the problem of operating systems, which form the interface between the very basic and difficult to use machine language of the computer and the more human-like abstract languages with which most applications programmers work. An applications program written for a particular operating system on a particular computer will not run with another operating system even for that same computer, let alone for another computer, unless the two operating systems have been specifically de-

\(^{104}\)See supra text accompanying notes 28-32.

\(^{105}\)Adaptors happen to be fairly cheap and not greatly inconvenient in the case of electrical appliances. This is not always the case for interface technology, however, as the situation in video recorders vividly demonstrates. Moreover, even if electrical adaptors are intrinsically cheap, a copyright in the shape of the plug would carry over to a monopoly on adaptors, so the monopoly profit attributable to third-party products adopting that plug shape would still be available to the copyright owner.
signed to be compatible, that is to say, so that the second will accept and properly execute all applications programs written for the first. Microcode raises the same type of problems, but at a more basic level inside the computer, namely, that of defining the fundamental instruction set and the timing with which such fundamental instructions are executed. Similar compatibility problems are now also beginning to surface at the other end of software-based machines. For example, communication protocols must be compatible if computers are to be linked together in an interactive network, and mobile telephones must be compatible with the retransmission stations that carry the messages along to their destination.

In all these cases, the interface between two machines or between programs and a machine consists at least in part of other programs, usually copyrighted programs. The copyright problem did not arise in the case of electrical plugs because machine parts are not copyrightable. It arises in the case of computer machine interfaces because programs, or at least some programs, are now copyrightable. The question is to what extent the copyrighted interface software can be copied for the purpose of achieving compatibility. Compatibility is obviously desirable for users, because it allows them to use applications programs interchangeably among computers and makes a larger number of applications programs available. Also, without compatibility, the dominant player in the market has a monopoly not only on his software but also on all software developed by third parties to run on his computer or system. The question is whether compatibility can be achieved, and achieved in a socially optimal way, without infringing copyrights in the original software.

1. Copying to Achieve Operating System Compatibility

The two Apple Computer cases both involved essentially direct reproduction of Apple's copyrighted operating system programs, so the findings of infringement were justified unless operating software is to be denied protection altogether. Both cases contain language and analysis, however, that is inappropriate to copyright protection of programs, and to operating systems in particular. The language shows a highly protectionist bent for computer technology that neither court attempts to justify on policy grounds relevant to software as a special kind of technology.

The Franklin opinion, in fact, in dictum expressly denies the desirability of third-party program compatibility as a factor in determining the scope of copyright protection in an operating system. The court also defines the idea of an operating system program according to its overall function, such as translating

---

source code into object code. In this the court erroneously applies to technology the broad view of expression appropriate to novels and plays even though traditional copyright law requires nearly verbatim similarity before infringement is found in technological "expression." Both courts embark on the false analogy to literary works in which the ideas contained can be expressed in a plurality of ways. Computer programs involve technological efficiencies, and neither court has a word about how its approach to protection would affect the development of technological efficiency. More recently, a district court dealing with mobile telephones rather than operating systems took an approach more consonant with the technological nature of programs, finding no infringement with respect to that part of software that was essential for compatibility with the plaintiff's system.

Most commentators agree that, at least to the extent necessary for compatibility, copying of another operating system should be permitted. The question of how much copying is necessary, however, is an empirical one for which we do not have sufficient data to clearly resolve the legal issues. One commentator has suggested that the existence of operating systems developed through "clean room" procedures and compatible with some of the most popular (and simpler) operating systems, like that for the IBM PC, demonstrates that there is never a need to copy any part of an operating system for purposes of compatibility; developers of competitive systems should therefore view the original as a "black box," available for study only at its input and output stages. There are, however, other popular operating systems that have not been cloned, such as that for the Apple Macintosh, and an officer of a minicomputer company testified in an antitrust case—against the interests of his company—that it would be impossible to develop operating software performing all of the functions of the operating system at issue without violating his company's copyright and using its trade secrets.

Moreover, we have only anecdotal data on the relative costs of developing

109 Franklin, 714 F. 2d at 1253.
110 Franklin, 714 F. 2d at 1253; Formula, 725 F. 2d at 525.
111 E. F. Johnson, 623 F. Supp. at 1502-03. Other aspects copied by the defendant were not necessary for compatibility, however, and the court there found infringement.
112 E. g., Davis, supra note 57; Franklin, Misappropriation Law, Copyright Law, and Product Compatibility, 4 COMPUTER L., No. 2, at 8 (Feb. 1987); Grogan & Kump, supra note 5; Raskind, The Uncertain Case for Special Legislation Protecting Computer Software, 47 U. PITT. L. REV. 1131, 1170 (1986).
113 Davidson, supra note 79, at 1077-78, 1080-82.
114 Digidyne Corp. v. Data General Corp., 734 F.2d 1336, 1342 n.3 (9th Cir. 1984). The defendant was charged with using the copyright in its popular minicomputer operating system to create an illegal tie. If the operating system could have been easily duplicated, the tie would have been weaker, so such testimony was against his company's interests.

A recent article in the Japan Economics Newspaper described Honeywell's new software approach to compatibility (Integrated Information Architecture), which is said to permit its computers to be used with IBM's as if the two were a single computer. This may be a significant technological advance toward compatibility, but it does require having both computers, as opposed to the traditional approach in which the compatible computer is intended as a substitute for the original.
compatible operating systems by "clean room" procedures. It seems possible that, in at least some cases, it might cost more than developing the original system itself.\textsuperscript{115} Clean room procedures involve waste and duplication solely for the purpose of proving independent effort,\textsuperscript{116} and they are inefficient from a technological point of view because they inhibit communication among the members of the development teams.\textsuperscript{117} Furthermore, any technological ideas in the original that are unprotected by copyright will not be available to the programmers operating in the "clean room." While "clean room" procedures can be a safe harbor for those subsequent developers who wish to use them, to say that independent development is the only defense to an infringement claim would be to extend the copyright protection for programs not only beyond traditional intellectual property law protection for technology but even beyond that for traditional copyrightable works.

On the other hand, if developers of compatible operating systems are permitted to look at the original, the infringement issue under the traditional framework becomes whether there is substantial similarity of copyrightable elements in the two systems. There are clearly constraints that will require at least some similarity in order to achieve compatibility,\textsuperscript{118} and other similarities may result from a particularly efficient way of performing functions in the original that the second developer cannot improve upon. Because efficiency is important in program technology, as in most areas of technology, subsequent program engineers should not be barred from learning from their predecessors unless there is some policy ground for extending protection over software technology beyond that afforded to other technologies. Studying the program and

\textsuperscript{115} Karjal, \textit{supra} note 103, at 63 n.35. There was testimony in the \textit{Data General} case that creating and testing a compatible operating system would require millions of dollars and years of effort. 734 F.2d at 1342. Also, the President of Phoenix Technologies, which successfully cloned the IBM PC BIOS through "clean room" procedures, has stated that they went to "tremendous expense" that involved "more than simply building a BIOS" to be sure they were within clear legal boundaries. Wilder, \textit{Phoenix Technologies joins the hunt for pirates}, \textit{Computerworld}, Feb. 23, 1987, at 15.


\textsuperscript{118} Grogan & Kump, \textit{supra} note 5, at 20 (for example, various entry points and memory locations); Davis, \textit{supra} note 57, at 16. Mr. Davis also points out the extraordinary difficulty of separating idea from expression in operating systems—and the heritage of twisting a copyright law drafted for human-comprehensible works and applying it to machine technology.
using its efficient aspects is not piracy as defined herein, so the piracy ground for extension of protection is not available.

If we accept, however, that efficient programming methods incorporated into the first operating system, as well as other methods or aspects whether or not efficient but necessary for compatibility, may be taken by developers of competing systems, we will find the courts flooded with testimony concerning how other, more efficient methods were available to the subsequent developer and that therefore her adoption of the same methods as the original should be deemed infringing. It will be very difficult for a court to sift through this testimony, both for veracity and for relevance. If programmers, like other engineers, are permitted to learn from others, it follows that they need not always be smarter, especially when the testimony about more efficient methods comes at trial, several years of technological development later. The possibilities available at the earlier time will always appear more obvious when viewed retrospectively. In the airplane wing example discussed above, 119 we do not punish the second developer because he missed an even more basic improvement possibility that was open to him.

Moreover, much of the original program may itself be a collection of public domain programs, which, if protected at all under traditional copyright, is protected as a compilation. If the court is doing its job in the substantial similarity analysis, it must determine not only which similarities resulted from compatibility and efficiency considerations but also which arrangements of public domain programs were taken for the same reasons. There is also a tendency with highly technical evidence of the type presented in software cases for the court to rely not only on the expert’s opinion on technical matters but to seek her opinion on ultimate issues like substantial similarity of expression as well. 120 What types of similarities should be deemed sufficient to constitute infringement is a legal judgment that the court should not delegate to a technological expert untrained in law.

These issues are not substantially different from those that arise in proving infringement in any program. We will return to them in the section dealing with proof of infringement, 121 in which it is suggested that we abandon the search for an infringement standard that depends on substantial similarity of protected elements and instead adopt one that focuses on whether the defendant’s activities substantially created the advantages of piracy.

119 See supra text accompanying note 99.
120 Both the district court and the Third Circuit in Whelan, for example, relied heavily on the “more credible” or “more persuasive” testimony concerning similarity of the plaintiff’s expert as compared to the defendant’s. Whelan, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1246-47 (3rd Cir. 1986), cert. denied, 93 L.Ed.2d 831 (1987); id., 609 F. Supp. 1307, 1316 (E.D. Pa. 1985).
121 See infra text following note 192.
2. Microcode

The protection of microcode under copyright raises many of the same problems as operating systems, with a few special wrinkles. There is a legitimate technical issue of whether microcode qualifies as a "computer program" as defined in the Copyright Act. It depends on the interpretation of the term "computer," which the Act leaves undefined. Microcode is software that is normally stored permanently inside the computer's central processing unit. It determines the basic machine language instruction set seen by the outside programmer, and it is that instruction set that determines what the machine can do and how it does it. If the microcode is changed, the machine becomes a different computer. Consequently, it is not unreasonable to view microcode simply as a part of the machine that, until additional instructions are added (an operating system, for example), does not cause the machine to do anything. Under this view, microcode would not be a computer program and would be wholly unprotected by copyright. Another possibility leading to the same result is that many microprograms are so simple and their sequencing so dictated by their function and hardware environment that they lack originality. Because the amount of effort that goes into the design of the microcode itself (the sequences used to call the chip hardware resources into action) is relatively small in comparison to that required for the design of the physical resources of the chip and its overall architecture, it might be sensible policy to adopt one or both of these arguments, which would leave the protection of chip technology wholly to the Semiconductor Chip Protection Act specifically designed for that purpose.

The court in Intel did not discuss any of these arguments in implicitly rejecting them by ruling that Intel's microcode was copyrightable. This alone justifies a fear that the court may not be sufficiently considering the technological nature of the subject matter with which it is dealing. Given the technical arguments in favor of treating microcode as a machine part rather than a copyrightable program, the court should have considered whether a policy basis existed for extending copyright protection to this form of technology. It appears that the piracy basis for the protection of programs in general is lacking under

\(^{122}\)"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." 17 U.S.C.A. § 101.

\(^{123}\)See Karjala, The Limitations on Protection as Program Works under Japanese Copyright Law, 8 Mich. Y.B. On Int'l Legal Stud. _____ (1987); Karjala, Protection of Computer Programs under Japanese Copyright Law, 8 Eur. Intell. Prop. Rev. 105, 106 (1986); Hinckley, supra note 88, at 35; but see Laurie & Everett, The Copyrightability of Microcode: Is It Software or Hardware . . . or Both?, 2 Computer L., No. 3, at 1, 7 (Mar. 1985) (arguing that even a single microinstruction can be a program, but not discussing the definition of "computer").

\(^{124}\)Karjala, supra note 123, 8 Eur. Intell. Prop. Rev. at 106; Laurie & Everett, supra note 123, at 7.


\(^{126}\)NEC Corp. v. Intel Corp., No. C-84-20799-WAL (N.D. Cal.), Partial Findings of Fact and Conclusions of Law. Of course, the Semiconductor Chip Protection Act does not protect the Intel chip in question because it was commercially exploited prior to July 1, 1983. Nevertheless, the Act is an important factor in the policy debate over the copyrightability of microcode.
the facts of the case.\textsuperscript{127} Moreover, even if an aroma of piracy is found, the court should consider whether this type of piracy is now controlled by the Semiconductor Chip Protection Act.

In determining whether NEC infringed Intel’s copyright, the court will have to examine whether any similarities in the two versions of the compatible microcodes are dictated by function and hardware,\textsuperscript{128} or were in fact arbitrary choices among a wide variety of equally compatible and equally efficient possibilities. Because the allegedly similar microprograms involved in the case are so simple,\textsuperscript{129} NEC has a good chance of prevailing on this issue. A basic policy issue, however, remains submerged when we look only at the range of programming choices available. Suppose, for example, that the evidence at trial shows that the NEC microcode design team “cheated” on some of the allegedly offending fifty-four microprograms, that is, they simply followed Intel’s choice without going to the trouble of rearranging the order or looking at whatever other possibilities existed that would achieve the same function. Even if taking verbatim an equivalent portion of a traditional literary work would constitute infringement, programs are literary works only in form. In substance they are technology, where we want to encourage innovation by cumulative improvement. The heavy-handed remedies of copyright, including outright injunction of infringing works, are too severe for activity that would not even be wrong in any other area of technology. Any copying of the type described would give NEC essentially no advantage in its future competition with Intel, because the amount of time saved would constitute a minuscule part of the development costs of NEC’s chip. The pejorative term “piracy” should not be applied here, even if there was copying. This issue, too, ties in with the general one of proving infringement and should be considered in the light of the analysis in that section.

\section*{B. User Interface Compatibility}

The user interface compatibility problem does not arise so much from the nature of programs as technology as it does from the fact that many programs are designed to interact with a user to perform their tasks. Part of that interaction is through mechanical devices such as a joystick, a mouse, or a keyboard,

\begin{itemize}
\item\textsuperscript{127}See supra text accompanying note 93.
\item\textsuperscript{128}Intel is apparently taking the position that, although the hardware was unprotected, NEC chose to use the same hardware to enable NEC to copy Intel’s microcode. Dunlap, supra note 88, at 18-19. But for the protective approach courts generally are taking toward the protection of computer software, one would dismiss this outlandish argument with derision, as it completely reverses the limited scope of copyright in useful articles. Instead of denying copyright to the extent that design is inseparable from the utilitarian features, it argues that implementation of the utilitarian features is infringement if something else exists that performs the same function.
\item\textsuperscript{129}According to NEC’s counsel, the average length of the microprograms claimed to be similar is slightly less than four lines of code. Hinckley, supra note 88, at 54. Compare this to the some 186,000 lines of code in a statistical analysis program. SAS Institute, Inc. v. S & H Computer Systems, Inc., 605 F.Supp. 816, 822 (M. D. Tenn. 1985).
\end{itemize}
with respect to which no one has yet asserted a claim for copyright protection. The images that appear on the screen, however, are easily (perhaps too easily) analogized to images on a printed page, and the level of protection they should be afforded under copyright is a matter of intense debate.

We must distinguish here program output over which the user has only very limited control, like a videogame. The user does interact with these programs through interpretation and manipulation of screen images, but user discretion is limited to the very narrow range of choices made available by the program for the purpose of playing the game. If the user tries to do anything else, either the game will not work or she will immediately lose an uninteresting contest. Moreover, videogames, like novels and movies but unlike word processing or spreadsheet analysis, are desirable more for their variety than for their uniformity. Consequently, protection of this type of output under copyright and by analogy to books or films is appropriate.

Many other types of application programs, however, produce output more appropriately analogized to the accounting system ruled uncopyrightable in Baker v. Selden than to literary or audiovisual works. Use of a spreadsheet program, for example, involves more than learning a series of keyboard operations or mouse movements; additional choices are placed directly on the screen, and still more are inherently contained in the format of the lines and boxes the program places on the screen. Anyone familiar with even a word processing program knows that becoming adept at its use takes some practice. Spreadsheet programs and statistical analysis packages probably require even more training before proficiency sufficient for efficient use is reached. Just as few of us have learned more efficient typewriter keyboards even though they are now available, most people are reluctant to retrain to learn new word processing or spreadsheet programs, even when newer models offer significant advantages over the old. They are “locked in” to the programs on which they were trained.

Standardization of these user interfaces promotes efficiency and reduces the need for undesirable retraining. The value of standardized user interfaces even across different programs is now apparent, and manufacturers like Apple are encouraging third-party programmers to adopt standardized user interface procedures. Moreover, no useful purpose is served by requiring differences simply for the sake of being different. In fact, copyright protection for these features could be disastrous. Once a particular program became established, competitors would have little incentive to make innovative improvements, be-

\[130\] Reback & Hayes, supra note 106, at 2.


\[132\] Siegel & Derwin, Copyright Infringement of the “Look and Feel” of an Operating System by its Own Applications Programs, 4 COMPUTER L., No. 1, at 1, 2 (Jan. 1987).

\[133\] Comment, Protecting the “Look and Feel” of Computer Software, 1 HIGH TECH. L.J. 411, 425 (1986).
cause the potential market could be greatly reduced by the large number of users locked in to the original system. And absent this competition, there would be reduced incentive even for the original programmer to make improvements.\footnote{It appears that this may have happened with word processing software. I personally know people in legal education who began word processing with the first popular program and have stuck with it long after programs with significantly advanced features became available. It took several years after the appearance of these advanced programs before the makers of the first one reacted with improvements incorporating the features offered by the competition. Even here, the reaction was probably only because these other programs did finally begin to attract much of the new market, which was still growing.}

Nevertheless, notwithstanding an early and well-reasoned case holding input data formats for a structural analysis program uncopyrightable,\footnote{Synercom Technology, Inc. \textit{v.} University Computing Co., 462 F. Supp. 1003, 1014 (N.D. Tex. 1978).} two district courts have now found aspects of user interface screen displays protectable under copyright.\footnote{Digital Communications Assocs. \textit{v.} Sofiklone Distributing Corp., 2 U.S.P.Q.2d (BNA) 1385 (1987); Broderbund Software, Inc. \textit{v.} Unison World, Inc., 231 U.S.P.Q. (BNA) 700 (N.D. Cal. 1986). These cases have already rattled the software industry. \textit{See} Schwartz, \textit{Software Industry Stalled by Suits}, Nat’l L. J., June 29, 1987, at 3.} 	extit{Broderbund} involved menu-driven programs used to design greeting cards and posters, while the programs in 	extit{Sofiklone} were used to put one computer in communication with another. Both courts fell prey to the false analogy to literary works in which a number of ways are available to express the desired idea. Having found the screen displays sufficiently complex that a wide variety of arrangements was possible, they concluded that the displays were protected expression.\footnote{Sofiklone, 2 U.S.P.Q.2d (BNA) at 1393; \textit{Broderbund}, 231 U.S.P.Q. (BNA) at 704. One commentator has similarly concluded that the utilitarian purpose of input formats is to enable the user to communicate with the program; because this purpose is separable from their sequence and arrangement, the sequence and arrangement should be protected. \textit{Note}, \textit{Defining the Scope of Copyright Protection for Computer Software}, 38 Stan. L. Rev. 497, 525-26 (1986). However, this author misses the standardization issue entirely.}

In 	extit{Broderbund}, the court found that the design of the audiovisual display was dictated primarily by artistic rather than utilitarian considerations, but that analysis could never be sufficient under traditional law to justify copyrightability. An electric battery, for example, could be designed to operate a mechanical doll as part of the doll’s external appearance, in which case the shape of the battery would be determined by artistic considerations. But the battery remains a utilitarian device that must take that particular shape if it is to be used for the purpose intended, and a copyright on the shape would give the doll designer a seventy-five year monopoly on the battery. The question with software is whether there is a danger of user lock-in, and useful programs always involve a danger that one will become very popular to the virtual exclusion of others. In that case, a copyright on the screen display is a virtual monopoly on the utilitarian article, namely, any program that performs the function in question. The 	extit{Broderbund} court did not consider this problem or the value of standardization of these interface techniques across users.
The court in Softklone did find that certain general aspects of the user interface were unprotected ideas, such as the use of a command-driven program and the typing of two symbols to activate a specific command. However, even the approach of highlighting two specific letters to indicate how to activate a particular command, and a fortiori the actual letters chosen to activate a particular command, were deemed protected.\textsuperscript{138} Even worse, the court expressly rejected the defendants' appeal to the value of standardization, misinterpreting the Whelan court’s correct statement that progress in computer technology is not qualitatively different from that in other areas of science or art—a misinterpretation that the Whelan court itself made.\textsuperscript{139} The inhibition of technological progress that would result is precisely why we do not give copyright protection to the layout of typewriter keyboards. For the same reason, we should not give copyright protection to user interface screens or formats.

Both Broderbund and Softklone therefore represent dramatic extensions of copyright protection beyond the traditional bounds, while neither case offers any convincing policy basis for such extended protection.\textsuperscript{140}

\textsuperscript{138}Softklone, 2 U.S.P.Q.2d (BNA) at 1391. Thus, according to this decision, the use of the letters SP, shown on the screen as “SPEED” with the first two letters highlighted, is the seventy-five year property of the plaintiff in the case. Yet the use of the first two letters of a particular command seems the most natural way of providing a reasonable number of short combinations for commands, a necessary feature of an efficient program.

\textsuperscript{139}See supra note 14 and infra text accompanying note 160.

\textsuperscript{140}West Publishing Co. v. Mead Data Central, Inc., 799 F.2d 1219 (8th Cir. 1986), is a software-related nonprogram case raising an unusual user interface issue. It, too, probably should be classified under a protectionist heading. As is well known, the plaintiff and defendant offer competing database services providing access to judicial decisions and other law-related source materials. The plaintiff is also the publisher of the standard hardbound case reporters and has a de facto monopoly in that area. The legal community has adopted the custom of citing decisions by volume and page number of many of the West reporters, and in citing an opinion for a specific point, lawyers often give the internal, or “jump” cite as well, enabling the reader to find the relevant point more rapidly. Because of this custom, the defendant sought to use a standard star pagination system that would indicate the West page breaks in the text presented on the monitor screens of its customers. This would be an obvious convenience for people using the defendant’s database service, because while researching for briefs or articles they would be saved the trouble of going to the library shelves to get the “official” cite.

The plaintiff agreed that citation to the first page of the cases published in its reporters could be used by the defendant. Id. at 1222. It argued, however, that the defendant’s star pagination system for succeeding page numbers infringed its copyright on the arrangement of cases, and the court held that the plaintiff had a likelihood of success on that claim. This case highlights dramatically the difficulties with the “selection and arrangement” rationale for the protection of compilations, see supra text accompanying notes 64-76, because the court had to do much fancier footwork to find originality in the plaintiff’s arrangement. (Clearly there was none in the selection, as all cases received are printed.) The real issue is whether the plaintiff’s de facto monopoly in the printed reporters, which the legal community adopted as its standard at a time when nothing else was available, should carry over to legal databases of this type. If the legal community retains its standard, as seems likely for reasons of inertia if no other, the plaintiff’s database service has a significant user interface advantage over the defendant’s, because the plaintiff can provide information necessary for internal citation but the defendant cannot (assuming the proposed star system infringes the plaintiff’s copyright).
C. The Scope of Copyright Protection in a Program

1. The Idea/Expression Distinction

A. INTRODUCTION

While vague, the idea/expression distinction has long been a fundamental part of copyright law. The broad scope of what constitutes expression in novels and plays and the narrow scope for historical and factual works represent policy judgments by the courts concerning what should be protected and what should be freely usable by subsequent authors. Until the advent of computer software, all copyrightable works were intended to and did communicate with a human audience, either directly or through a machine like a phonograph. Judges therefore felt reasonably comfortable in making the necessary policy distinctions appropriate to the different types of works with which they were faced.

The decision to protect programs under copyright meant that courts would inevitably begin looking for standards by which to apply the idea/expression distinction to programs. The problem is that programs are not intended to communicate with anything or anybody; rather, they are intended to operate computers. Because computers execute instructions sequentially and mechanically, the human conceptual separation of idea from expression has no meaning for a program running on a computer. The computer has no conception that it is carrying out, say, a regression analysis when someone combines data with a statistical package and pushes the appropriate keys at his terminal.

Nor does the idea/expression distinction make any sense from the point of view of the user. The user of a program does not even see the individual lines of code and interacts with the program only through its output, which is wholly separable from the program itself. The user takes the program as a whole and uses it as a whole, in a computer. She cannot and does not separate it into idea and expression.

We have therefore two choices: First, we can abandon the idea/expression distinction as inappropriate to this radically different type of copyrightable work. This choice would mean we must find some other policy-based standard for determining what aspects of a program, if any, can be taken by others and what aspects are protected. I argue below that this path provides a surer and clearer means of effecting the socially desirable level of protection for software technology.

Second, we can attempt to give the idea/expression distinction some meaningful content in the program context. This means we cannot adopt the distinctions used for traditional works without making adjustment for the important differences arising from the nature of programs as technology. Rather, we must find a standard that makes sense of the distinction between idea and ex-

---

141 See supra text accompanying notes 47-58.
142 See infra text accompanying note 178 and text following note 192.
pression in a program while aiming for a socially optimal level of protection. Because most courts and commentators have tried to apply the idea/expression distinction to programs, I will consider this choice first.

B. THE IDEA/EXPRESSION DISTINCTION
   AND SUBSEQUENT AUTHORS

   We have seen that the idea/expression distinction has no meaning from the point of view of the user or the computer. There are, however, people who do read programs, namely, other programmers (including users who wish to modify a program). Some of these other programmers will be employees of the company that developed the program, who will read it for the purpose of making improvements and repairs. The scope of protection is rarely an issue for these programmers. Programmers unrelated to the copyright holder, however, may also wish to read the program for the sake of learning from it, perhaps with a view to writing competing programs. Focusing on this group seems appropriate for applying the idea/expression distinction, because it is precisely with a view to the relative rights of the original author and subsequent competing authors that copyright law has drawn its traditional balances.

   When idea/expression is approached from this angle, the fundamental nature of programs as technology leads first to the conclusion that infringement should be limited to verbatim or nearly verbatim copying, as is appropriate for traditional works of fact.\textsuperscript{143} This would be sufficient to prevent direct piracy through slavish copying of source or object code. It appears, however, that relatively simple, but still essentially mechanical, operations can effectively disguise activities that could be regarded as piracy as defined herein—taking a meaningful portion of another program without analysis and using it or putting it into competitive distribution without substantial development costs.\textsuperscript{144} We must therefore go beyond literal or near-literal copying if protection is to be meaningful.

C. THE BLACK BOX APPROACH

   One commentator has suggested that everything in a program be protected except its function as determined from input/output analysis, that is, that the program be considered a "black box" at the insides of which subsequent programmers are not permitted to look.\textsuperscript{145} Reverse engineering of programs by looking at program structure or details would not be permitted. He admits that this would constitute a "controversial" change from what is permitted in other fields of technology but claims that wheels are easily reinvented in the program

---

\textsuperscript{143}See supra text accompanying notes 57-58.

\textsuperscript{144}For example, one might decompile a proven program into its source code and then recompile that source code with a different compiler.

\textsuperscript{145}Davidson, supra note 79, at 1080-82. Later Mr. Davidson treats the black box approach more as a safe harbor rather than a firm standard for distinguishing idea from expression, id. at 1082, but he also seems to believe that the derivative work concept creates something of a presumption of infringement once the black box has been penetrated. Id. at 1093-99.
context and that even a "peek" inside the black box can reduce to weeks what would otherwise take months or years to develop independently.\textsuperscript{146} The policy basis for this approach is said to be to reward above-average software whose function is difficult to duplicate and to force competitors to design in their own ways, thereby encouraging innovation and improvement.\textsuperscript{147}

Neither of these grounds for deviating from our traditional approach to technology protection has an empirical, or even a plausible theoretical, foundation. The essential difference between software technology and other types of technology is that machine design can be accomplished with human-symbolic representation.\textsuperscript{148} This does make copying and competition based on that copying easier, and it calls for steps to control activities that would otherwise destroy the incentive to create. But it does not mean we can change human nature or the manner in which technology develops. As argued above, programmers do look at what other programmers do, and socially valuable improvements in program technology can be and are made by incremental additions to the existing base. There is no evidence for the proposition that software "wheels" are any easier or more difficult to invent than any other type. And there is utterly no reason to think that making every programmer start from scratch would result in innovation or improvement. Programmers use the technological methods they learn in school and subsequently develop on the job, just as any other type of engineer. But what engineers learn on the job, and even what they learn in school, is partially the unprotected current knowledge developed by others working in similar fields. To deprive programmers of this way of improving their skills could easily impose a heavy social cost, as programmers would learn little after leaving school except what is developed, always independently if people are obeying the law, at their respective places of employment.

D. STRUCTURE AND ORGANIZATION

1. The Cases

Rejection of the black box approach thus assumes that programmers will and should be able to look at the programs of others for the purpose of extracting technological ideas. It remains, however, to describe what we mean by an idea in this context, and that is by no means an easy task. Most of the debate has centered around what has come to be called program organization or structure. Should program structure be considered protected expression or should it be considered unprotected idea? In fact, the question cannot be sensibly addressed without bearing in mind throughout the analysis both the correct analogy of programs to utilitarian technological works and the antipiracy basis for affording special copyright protection to program technology over other technologies.

\textsuperscript{146}Id. at 1081, 1097.
\textsuperscript{147}Id. at 1098.
\textsuperscript{148}In an earlier article, Mr. Davidson appears to have recognized this point. Davidson, supra note 1, at 342-45.
The strong protectionist trend in the cases began with an unnecessary dictum in Franklin. Essentially slavish copying was admitted, so the court could easily have found infringement without commenting on the scope of protection, but it devoted a section of the opinion to the idea/expression distinction and essentially adopted the black box approach described above: The idea of a program is its function, and if that function can be achieved in a variety of ways, the particular way chosen is protected expression. It did not consider the possibility that some of those ways might be more efficient than others, or the social cost of requiring practitioners of this new technology to do everything from scratch rather than building on the existing technological base.

The real villain, however, is the much discussed Whelan case. Here the defendant used the plaintiff’s source code program for managing a dental laboratory as a basis for writing a program that would perform the same function on a different computer. No evidence of literal code similarity was presented, so the question was whether similarities in file structure and five particular subroutines were sufficient to uphold the lower court’s determination of infringement. As is now well known, the court held that copyright protection extends to a program’s structure, organization, and sequence.

The Whelan court’s long step in the direction of copyright protection for technology was not taken in ignorance of the fundamental nature of programs. Indeed, the opinion shows a good understanding of program technology. Unfortunately, the court forgets everything it has learned as soon as it applies copyright law to the facts before it. At that point the analysis shifts to the now familiar, but wholly inappropriate, analogy to traditional literary works. As a result, the court supplies no policy basis for the dramatic shift it makes in the traditional level of protection for technology.

---


\(^{150}\) Id. at 1253.

\(^{151}\) Whelan, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3rd Cir. 1986), cert. denied, 93 L.Ed.2d 831 (1987).

\(^{152}\) The district court found that the defendant’s action could not even be compared to the translation of a novel from English into French, because of the very different hardware environments in which the programs were to operate. Whelan, 609 F. Supp. at 1321-22.

\(^{153}\) The opinion also appears to approve the lower court’s reliance on similarity of screen output as probative of program similarity. 797 F.2d at 1243-45. Fortunately, the court does not seem to rest its determination of infringement on such nonsense, which was rejected even by the protectionist opinion in Digital Communications Assocs. v. Softklone Distributing Corp., 2 U.S.P.Q.2d (BNA) 1385, 1388 (1987) (copyright in a program does not extend to the screen displays, which are separately copyrightable, so copying screen displays in itself does not support a claim of program infringement). To the extent there is any truth in the proposition, it strongly argues against a finding of infringement: the issue is the substantial similarity of programs, not of output. If the output is unprotected, for example, any similarity in the program necessary to achieve that output must also be unprotected. Otherwise, copyright affords a monopoly in a utilitarian article, namely, computer programs that achieve the output in question. But if the output alone is evidence of expression in the program, such expression must be, to that extent, necessary to achieve the output. Consequently, idea and expression merge to the same extent, and copyright protection must be denied.

\(^{154}\) 797 F.2d at 1248.
Early in the opinion the court recognizes that programs involve technological efficiencies that make them more or less valuable. It equally recognizes that these efficiencies depend in large part on the arrangement of modules and subroutines, that is to say, on the program's structure. Data organization can also involve efficiencies that make the program more or less valuable. Yet the opinion nowhere considers the technological value of building on the accomplishments of others. It does not even consider the possibility that the particular similarities taken were the best the defendant was able to do under the circumstances or that the plaintiff's design of those particular features was among the more efficient ways of accomplishing the desired result. Nor does it consider whether the methods adopted in both programs were natural or obvious ways to write a program for managing a dental laboratory or dictated by their common goal.

In determining the standard for distinguishing idea from expression, the court misreads Baker v. Selden as supporting the proposition that a variety of methods for achieving an end implies that each method is protected expression. Baker did not look to whether there were other methods of keeping books but only to whether a copyright in the charts would monopolize the particular system developed by the plaintiff. Even more important, in rejecting the Whelan defendant's argument that structural similarity could not be shown by comparison of only a small fraction of the two works (five subroutines), the Whelan court turned to the literary work cases holding that taking a qualitatively important portion of a work can infringe, even if it is quantitatively relatively small. This approach ignores the vital differences between technological works and other works. Innovative technological developments are not protected by copyright even if they are of great importance. In fact, the more important they are, the stronger the policy basis for denying copyright protection.

While the Whelan court recognized that progress in computer technology was not qualitatively different from progress in other areas of science or the arts, it failed to discern the implications of this observation for a determination of infringement based on similarities of small pieces, even if important

---

155Id. at 1230.
156The court did consider the possibility that the similarity in file structures was dictated by efficiency, simply upholding the lower court's finding of fact that this was not such a case. Id. at 1243 n.43. Nevertheless, such a finding would not be sufficient to find infringement under traditional law. Baker v. Selden did not deny copyrightability on the ground that there were no other ways to do accounting (efficient or inefficient); it denied the copyright claim on the plaintiff's particular system. Wharton, Use and Expression: The Scope of Copyright Protection for Computer Programs, 6 Computer/Law J. 433, 462-63 (1985). See supra text accompanying notes 38-39.
157Wharton, id. at 462-63.
158Whelan, 797 F.2d at 1245-46.
159For example, the most important aspect of Einstein's special relativity paper was probably his assumptions about the constancy of the speed of light and the formal identity of the laws of physics in two coordinate systems in uniform relative motion. Everything else, including the famous formula $E = mc^2$, follows from these assumptions. No one would say that Einstein's copyright in the paper would protect those assumptions.
159Whelan, 797 F.2d at 1238.
pieces, of the programs. Progress in technology, including computer technology, is cumulative. The defendant started with the plaintiff’s contribution and made a socially useful addition by enabling others to use the technology. If the defendant made significant original contributions, as seems possible if similarities existed only in parts of the programs, his activity would never be called piracy in any other area of technology (except possibly by the original creator). Particularly in the conversion of a program to run on an incompatible computer, the creator of the original program has a long headstart over competitors, because he is most familiar with both its structure and its details.

It may even be that conversion to run on a particular line of computer would be unprofitable without building on the existing base, because that particular computer is not widely owned. By providing copyright protection for the program structure, owners of such computers must rely solely on the judgment of the original creator concerning whether a conversion is worth the effort. In deciding that such users must wait seventy-five years before knowing whether they will be able to use the technological advance represented by the original program, the court should have considered whether there was a policy basis outweighing these factors that justifies so much more protection for program technology than for other technologies.

In fact, the Whelan court wholly ignored these considerations. It found a policy basis for protecting program structure and logic in the significant cost attributable to their development. Under traditional copyright, the cost of developing a utilitarian article is irrelevant; the utilitarian aspects remain unprotected. If development costs are relevant to software technology, it must be through the ease of copying that substantially eliminates the headstart innovators in other fields have over competitors, or that reduces competitors’ production costs to permit price cutting before the original creator has had a fair opportunity to recoup his investment. The defendant in Whelan argued that when programs are not copied literally, learning their overall structure and incorporating that structure into a new program to perform the same function can require almost the same amount of time as that required to create

---

161 The original developer of innovative technology, of course, is likely to label as “piracy” any use not expressly authorized by her. Simply that individual creators would like more protection in a particular case, however, does not show that we have been wrong all these years in denying more protection to nonpatented innovations.

162 To the extent there is no such headstart, the value to anyone in looking at the program is also reduced. We should then ask why we are getting so excited about protection rather than simply allowing competition to work in the usual way.

163 Whelan, 797 F.2d at 1237. Earlier the court included debugging, documentation, and maintenance in addition to structure and logic development as greatly outweighing the coding process in overall development cost. Id. at 1231. Debugging, documentation, and maintenance, however, are necessary for the converted program as well. These costs should be considered when piracy is recognized as the basis for special protection of programs.
the original. The court rejects this argument partially on the ground that the amount of effort expended by the copier is irrelevant to copyright infringement. Thus, the court uses economic cost as a basis for greatly expanding the notion of copyrightability but rejects similar costs in determining what constitutes infringement.

Moreover, the defendant's program in Whelan did not compete with the original, because it was designed to run on an incompatible computer. Consequently, regardless of how much the defendant's efforts were simplified by reference to the plaintiff's program, the defendant would not be undercutting the plaintiff's opportunity to recoup her investment in development costs. Of course, the defendant's program would be competitive with any program subsequently developed by the plaintiff aimed at the defendant's market, but here the defendant has no special advantage, because the plaintiff also has knowledge—better knowledge than the defendant—of the original program while creating the second.

Again, we must be wary of the analogy to traditional works. We reserve to the copyright holder in a novel the right to decide if, when, and how to make the novel into a movie. The scope of protection in novels is wide, because we are more interested in creating an incentive for a variety of different types of works. The scope of protection in technological works has always been narrow, because we want to let technology develop in its customary cumulative way. Society benefits most when everyone is permitted to share in technological advances. With a variety of mutually incompatible computers on the market, society gets a significantly increased benefit from allowing many people to compete to make programs available for computers with which the original is incompatible. It would behoove those who wish to deny this social benefit to point to a countervailing policy basis for their position. And if that basis is simply a desire to maximally reward the creator of the original program, it should include plausible differences between program technology and other technologies to justify rewarding programmers more than innovators in other technological areas.

The effect of the Whelan decision, like the Franklin dictum, is to adopt the black box approach to the idea/expression distinction. If similarities in a relatively small part of the structure are sufficient to infringe, it is difficult to imagine anything below the level of general program function that could constitute unprotected idea. Three other cases have considered the conversion of programs to run on other computers. Two district court cases prior to the Third

\[164\text{Id. at 1237 (citing Note, supra note 13, at 1290). The Whelan opinion's first ground for rejecting this argument is the bald statement that it is not true that "approximation" of a program without verbatim reproduction is valueless. The defendant did not argue, however, that it was valueless, but only that it was costly. To the extent there is a basis for the defendant's claim, piracy as defined herein ceases to be a problem.}

\[165\text{See supra text accompanying notes 47-53.}

\[166\text{Indeed, the Whelan opinion states that the idea in the program was the efficient organization of a dental laboratory. 797 F.2d at 1240.}

FALL 1987
Circuit decision in Whelan split on whether structure and organization were protected, \(^{167}\) while the Fifth Circuit recently refused to follow Whelan in upholding the lower court's denial of a preliminary injunction on the ground that the similarities appeared, on the basis of the undeveloped record, to have been dictated by externalities. \(^{168}\) Although the Fifth Circuit decision is encouraging, the influence of Whelan has already been strong, \(^{169}\) and it is likely to remain strong until courts fully recognize the relevance of the technological nature of programs to the appropriate scope of their protection under copyright.

2. The Structural Test

Various approaches to the idea/expression distinction, in addition to the black box approach outlined above, \(^{170}\) have been suggested by the commentators. One commentator plumbs for a so-called “structural test,” that looks for similarities in the overall design of the programs and the interrelationship of their parts. \(^{171}\) While presciently predicting the Third Circuit decision in Whelan, this commentator does not consider the policy implications of the differences between programs as technology and traditional works. In fact, she bases her argument on the traditional protection given to patterns or themes of a novel or play. \(^{172}\) Consequently, as she comes to essentially the same conclusions as Whelan without offering any further justification for the result, her approach is subject to the same criticisms leveled above at the Whelan decision.

3. Commercial Competition

The “commercial competitiveness” approach has been suggested by Professor Raymond Nimmer and Patricia Krauthaus as a way of separating idea and expression when literal copying is absent. \(^{173}\) They would look to comprehensive transcription of a complex product and not merely selective takings of portions of the technology. Moreover, the result would have to be directly competitive with the original in a market that the original developer is likely to enter, and the transcription must represent the essential core of the two products, thereby creating “direct and pervasive” competition based on the original developer’s work.

Requiring comprehensive transcription rather than selective takings recognizes the need for program technology to develop sequentially and cumulatively.


\(^{168}\) Plains Cotton Cooperative Ass’n v. Goodpasture Computer Service, Inc., 807 F.2d 1256, 1262 (5th Cir. 1987).

\(^{169}\) For example, both Broderbund and Softkline relied heavily on Whelan in concluding that user interface screens were protected by copyright. See supra text accompanying notes 137-40.

\(^{170}\) See supra text accompanying notes 145-48.

\(^{171}\) Note, supra note 137, at 526-30.

\(^{172}\) Id. at 527.

\(^{173}\) Nimmer & Krauthaus, supra note 5, at 48.
and is therefore a valuable contribution to the ongoing colloquy on the scope of protection in software. However, the "essential core" of the programs may include basic programming ideas and methods to which the expansion of copyright protection requires justification in policy. Moreover, the emphasis on competition unrelated to piracy is difficult to understand. In all other areas of technology we want to encourage competition in ideas. We should wish to do the same with program technology, subject to the special antipiracy considerations that distinguish programs from other fields of technology. Looking at whether the two programs compete does not address the problem of whether there was piracy—copying that substantially eliminates the traditional headstart for the original creator or substantially reduces development costs for the copier that permits undercutting price before the original creator has had a fair opportunity to recoup her investment. As discussed above, neither condition is satisfied when a program is converted to run on a computer with which the original program is incompatible. And when antipiracy considerations do not justify a change from traditional levels of technology protection, we must ask for specific policy grounds for extending protection and thereby decreasing competition that, in any other area of technology, has always been considered desirable.

4. Levels of Abstraction

The "abstractions approach" of Messrs. Reback and Haynes starts from the oft-quoted Learned Hand description of the idea/expression distinction in the context of literary works and thoughtfully applies it to programs. These authors divide a program into various levels of abstraction: the function or program level; the module and subroutine structure levels; the algorithm level;

---

174 See supra text accompanying notes 163-65.
175 As a technical matter, the commercial competitiveness approach also seems to confuse infringement and fair use issues. We look to competition as an important factor in determining whether an otherwise infringing taking is a fair use. 17 U.S.C.A. § 107(4). But determining whether there is infringement in the first place requires us under the traditional analysis to separate idea from expression, because no one is protected from competition in ideas. Commercial competitiveness is a distinguishing factor in the "sweat of the brow" cases, but they are not properly analogized to computer programs. See supra text accompanying notes 59-76. It is at best an indirect factor in the expanded notion of expression in novels and plays that include elements of plot, and in any event, these works are not properly analogous to programs either. See supra text accompanying notes 49-58. Protection is denied to theories and the fruits of original research in historical and scientific works, notwithstanding that those elements may be the crucial factor in any commercial success such a work enjoys. Hochling v. Universal City Studios, Inc., 618 F.2d 972, 980 (2nd Cir. 1980), cert. denied, 449 U.S. 841 (1980); Nimmer & Krauthaus, supra note 6, at 34-36; see also supra note 159.
176 Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2nd Cir. 1930), cert. denied, 282 U.S. 902 (1931).
177 Reback & Hayes, supra note 106, at 5-7.
and the code level. They argue that because idea/expression is a basic policy decision, we should evaluate the author’s contribution at each level and weigh against it the cost to society of granting a monopoly over that contribution. Because protection at a high level of abstraction can foreclose competition, a substantial contribution should be required for protection at that level, and even contributions at the lowest levels should not be protected to the extent they are necessary or dictated by externalities.

The major problem with this approach, in copyright terms, is that it focuses on creative contribution as the object of copyright protection. Thus, these authors argue that IBM’s choice of entry points for its BIOS operating system was arbitrary rather than creative and that copyright protection would give a patent-like monopoly over third-party applications programs. Therefore, the choice should be treated as an unprotected idea. But what if the choice had been creative and the same monopoly resulted? Under the traditional abstractions approach, exactly where to draw the line between expression and idea is hazy, but once drawn everything below the line is protected and everything above is freely available. In application to programs, however, as advocated by Messrs. Reback and Haynes, the line is in fact never drawn. They imply that protection is available, if sufficiently creative, at even the highest (program function) level of abstraction (presumably for the full period of copyright), and they explicitly withhold protection at the lowest levels if the particular code is dictated by necessity or externalities. The level of abstraction enters the analysis only in counterbalancing how much creativity is required for protection. Thus, they end up protecting technological creativity, a very new function for copyright law. Their approach fails to relate the level of protection afforded to the anti-piracy policy underlying program protection, and it supplies no other policy basis for protecting creative program technology more extensively than creativity in other technological areas.

E. SUMMARY

Again in the idea/expression area, the same theme arises: copyright has never protected technological creativity any more than it has protected against technological competition. To do so would be a radical change from the traditional balances copyright has drawn. There is no evidence that Congress intended such a radical change when it adopted the suggestion of the CONTU Report and explicitly extended copyright protection to computer programs. If we are to make this radical change through judicial interpretation, we should be clear about what we are doing and the reasons for doing it. And we should ask whether the full panoply of copyright protection should come along with the change. For example, if conversion of a program to run on an incompatible computer is deemed an infringement, should we at least be considering some system of compulsory licensing in an effort to encourage as much extension of computing power as possible in the most efficient manner? And if we are to grant a monopoly in significantly creative technological
developments, are we sure that we want the monopoly to continue for the full seventy-five year protection period of copyright?

If we are to retain the idea/expression distinction in the case of programs and at the same time avoid too radical a departure from the restricted levels of protection offered to other technological innovations, we must have a test that permits the taking of technologically efficient methods for achieving a particular result in a program but at the same time protects against the unfair advantages of piracy. Theoretically, we could have a test that considers whether each similarity in the two programs was a standard part of the technology available to all, or was in any event taken by the defendant because it was compelled by externalities or in the interests of efficiency. In fact, however, even the most thoughtful judges are having difficulty in sorting through all of the technical detail that such an approach to idea/expression requires. In addition, as a matter of policy we must ask whether the results obtained justify the resources necessary to prove, in trial after trial, just what was standard and what was necessary for efficiency or compelled by externalities.

Moreover, in the final analysis, an idea/expression test based on externality or efficiency considerations raises more fundamental issues. It is true that there will often be a wide variety of ways of achieving the same result in a program, perhaps even with roughly equal efficiency. Nevertheless, programs are written to be used— to cause a machine to function in a particular way. Except for mistakes, nonfunctional comments, and deliberately laid traps, everything in the program is there for a utilitarian reason. Even assuming the dubious proposition that courts can separate in a meaningful way those program features that are uniquely efficient, in which idea and expression merge, from features that are roughly equal in efficiency to a wide range of other possibilities, this approach to the scope of protection meshes only imperfectly with the policy basis for software technology protection. For one thing, it tends to protect the less efficient aspects of programs rather than the most efficient, which is anomalous. If copyright protection is extended to cover utilitarian features, limiting the extension to relatively inefficient features is justified only by the negative notion that it does less harm than full-scale protection of technological efficiency. Furthermore, while this approach partially addresses the piracy problem by requiring subsequent program authors to rewrite wherever rewriting is feasible, it goes too far because it requires rewriting for the sake of rewriting, even in cases where piracy is not a problem. For example, when the subsequent author attempts to convert a program to run on a computer with which the original is incompatible, piracy in the sense relevant to software technology protection used in this article178 is not an issue. By requiring such rewriting, we hamper the efforts of subsequent programmers to spread innovations in program technology rapidly and widely throughout society.

Perhaps sensing at least some of these issues (most likely the difficulty of

178 See supra text accompanying notes 163-65, 174.
separating the uniquely efficient from the widely available and equally efficient), some courts have implicitly abandoned the idea/expression distinction in favor of more direct standards of infringement, and they have support from some of the commentators. While this article does not agree with the specific means chosen for abandoning idea/expression in the program context, it offers another approach that more closely parallels traditional levels of protection for technology without importing the complexities of proof that adherence to the idea/expression distinction would entail. The next section covers these more direct approaches.

2. Proof of Infringement

A. THE DIRECT COPYING APPROACH

The direct copying approach drops the substantial similarity test altogether (not just the idea/expression distinction) and looks at the defendant’s conduct. In particular, because convincing evidence of copying is so often available, the traditional reliance on access plus substantial similarity is unnecessary. Rather, if there is evidence of copying that results in economic benefit to the defendant, the court should find infringement.

The questionable relevance of economic benefit to the defendant, unrelated to piracy, in determining copyright infringement was discussed above in connection with the commercial competition approach and need not be repeated here. To treat copying alone as sufficient for infringement is to make copyright a general misappropriation law for programs, one that prohibits takings of any kind from the program of another (except as tempered by the economic benefit condition). To argue on technical grounds that substantial similarity is merely a way of proving copying is to miss the basic point that infringement under traditional law arises only when there is copying of expression. That is, substantial similarity of protected features is a substantive element of the claim. Moreover, there remains the more important policy issue of why such a vast extension of protection should be made available specifically to computer program technology. Without dealing with this basic policy issue, some courts have at least implicitly adopted the direct copying approach, again showing the protectionist trend of the cases.

---

179 Conley & Bryan, supra note 13.
180 See supra text accompanying note 79.
181 Conley & Bryan, supra note 13, at 114.
182 See supra text accompanying notes 173-75.
183 Goldstein, Infringement of Copyright in Computer Programs, 47 U. Pitt. L. Rev. 1119, 1126 n.29 (1986); Hinckley, supra note 88, at 52 n.122.
184 E.g., M. Kramer Mfg. Co. v. Andrews, 783 F.2d 421, 445 (4th Cir. 1986). Similarly, the court in SAS Institute, while finding pervasive structural similarity, seemed heavily impressed by the obviously slavish copying of 44 lines out of some 186,000 lines of code and by the duplicity with which the defendants obtained their copy of the source code from which they worked. SAS Institute, Inc. v. S & H Computer Systems, Inc., 605 F. Supp. 816, 825-26, 830-31 (M. D. Tenn. 1985). See also the discussion of the E. F. Johnson case, infra notes 189-92 and accompanying text.
B. THE ITERATIVE APPROACH

The iterative approach to proving infringement in the computer program context has considerable appeal. Under this approach, the plaintiff has the burden of showing use of her copyrighted work by offering evidence of access and similarities. As discussed above, convincing evidence of direct use is often available in program cases in the form of identical errors, duplication of intentionally laid traps in the plaintiff's code, and similar common features. Next, the plaintiff must show that the defendant's program is an iterative reproduction, that is, a literal copy or translation of a substantial portion of the plaintiff's program. Then the burden of showing independent production shifts to the defendant.

This test is deliberately designed to meet the major difference between programs and other copyrightable works, namely, that programs are technology, not merely descriptions of technology, and that innovations in this technology as in all others are sequential and cumulative. Certainly it drastically limits the scope of protection in programs. If the plaintiff cannot show that the defendant's program contains the equivalent of a line-by-line translation or copy of a substantial part of the plaintiff's program, he is out of court. This is perhaps one problem with the approach. It addresses the piracy problem as if it arises only through literal copying or translating. If programmers discover other ways to eliminate the advantages now available to technological innovators in other fields, this very limited approach to piracy will not stop them.

Moreover, notwithstanding the narrow range of illegal copying envisioned by the iterative approach, it may nevertheless be overboard in some respects. The reason is that it focuses only on overall similarity (of substantial portions) and makes no provision for similarities that relate to technological efficiencies or external constraints. Possibly the author of this approach intended these factors to be taken into consideration in the independent production defense, but in traditional terms it is not independent production if the defendant actually copied. Couple this with the wide berth traditional copyright law has given to the concept of how much taking constitutes "substantial" and we might find this approach protecting much more than its creator intended.

For example, the court in E. F. Johnson purported to apply the iterative approach in a thoughtful opinion that nevertheless did not require anything like a showing of literal copying or translation of a substantial portion of the plaintiff's program. The court first found convincing evidence of direct copying, which now seems so common in software piracy cases: use of an identical sam-

\(^{184}\) Note, supra note 13.
\(^{186}\) Id. at 1288-94.
\(^{187}\) Drysdale, supra note 12, at 95.
\(^{188}\) See, for example, the discussion in Whelan emphasizing the qualitative rather than the quantitative approach to substantial similarity, citing some of the traditional cases finding infringement when only a relatively small portion of the original work was taken. Whelan, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1245-46 (3rd Cir. 1986), cert. denied, 93 L.Ed.2d 831 (1987).
pling rate when more speed was available with the newer and faster chip that the defendant was using; use of an identical sample error technique efficient for the plaintiff’s chip but inefficient for the defendant’s; exact duplication of an ‘‘H-Matrix’’ when any one of thirty-two forms would do equally well (together with the presence in both programs of an unnecessary inverse matrix); identical superfluous instructions in both programs; and an identical error that blocked transmission under certain circumstances. However, the court accepted these findings as evidence of substantial similarity rather than simply as evidence of copying. Under the traditional analysis, at least, the court should have gone on to analyze whether there was substantial similarity of copyrightable elements.

The defendant tried to raise this issue in two ways. It first argued that much of the plaintiff’s software consisted of uncopyrightable public domain materials. Next, it argued that in any event only ideas were taken. On the first issue, the court found that the plaintiff had indeed started from textbook matrices and tables but had added its own permutations and rearrangements sufficient to result in a more than trivial deviation from the public domain materials, which would justify a copyright. On the second issue, the court followed the Apple Computer cases to the effect that if there is more than one way to accomplish a task, verbatim duplication is infringement. The court concluded that the compatibility goal (with other mobile telephone equipment) could have been achieved without the verbatim copying found and therefore that expression was taken.

The fact that the defendant in this case copied verbatim some aspects of the plaintiff’s software that made the defendant’s system less efficient and failed to take advantage of the defendant’s better hardware lends a strong air of piracy to the facts and may justify the result. The defendant’s engineers could not have been thinking too hard about what they were taking, or why, and still have made all those mistakes. The approach taken by the court, however, boils down to finding infringement simply from copying, without analysis of whether the parts taken are protected by copyright. Certainly the defendant’s code was neither a literal copy nor a translation of the plaintiff’s, which means the court was not following the iterative approach as originally proposed. If what was taken was a reasonable technological solution to a technological problem, traditional copyright would not protect it, even if the existence of better ways of doing it are proved. Conversely, if what was taken verbatim was a relatively trivial part of the original software compared to a significant amount of independent development, as seems possible, for example with the defendant’s failure to rearrange the rows and columns of the matrices involved just for the sake of being different, there is no longer any policy justification for copyright liability, because there has been no piracy.

190 Id. at 1493-97.
191 Id. at 1501-03.
192 Under traditional copyright, while common errors are viewed as strong evidence of piracy, 3 Nimmer on Copyright § 13.03[C], even copying of a ‘‘factual’’ error does not prove infringe-
C. A SUGGESTED RESOLUTION

We need a way of protecting computer programs that recognizes the utilitarian nature of programs as works of technology for which the traditional policy of very limited protection outside of patent law is *prima facie* appropriate. At the same time we must recognize that the ease of copying and redistributing programs upsets the traditional balance between original creators' and subsequent improvers' rights by substantially eliminating the advantages innovators in traditional technologies enjoy by being first in the marketplace with a commercial product. Unless that imbalance is redressed, there will be little incentive to invest heavily in the development of useful programs for wide distribution.

While it seems possible to ask the courts to look for evidence relating to efficiency of operation and form dictated by externalities on the theory that those are unprotected ideas, that approach seems likely to engulf courts in waves of expert testimony on both sides, all of which is only peripherally related to the basic problem. The idea/expression distinction comes down to us from traditional copyright law, but traditional copyright does not protect technological innovation at all, whether it is highly creative or only slightly creative, and whether there is only one way to accomplish the desired result or many. We have decided to place programs under copyright protection primarily because copyright protection arises automatically and without formalities and expensive examination procedures. But the policy issues for software, and programs in particular, are different from other works, so it seems best to break away from the idea/expression distinction altogether to focus on those underlying policies.

Once we do that, the appropriate protective scheme begins to come into view. It is a variation of the iterative approach discussed above, modified at the second stage to meet the piracy problem rather than look for independent production. Slavish verbatim and near verbatim copying is infringement. That alone solves a good bit of the problem. The next problem is how we deal with charges that a nonverbatim copy or translation should nevertheless be held to infringe. First, we ask the plaintiff to produce evidence that the defendant made use of the plaintiff's work, either through "surprising" similarity of even a few code lines that contain errors, nonfunctional "traps," and the like, or through a showing of comprehensive similarity of structure that is unlikely to arise except through copying. When this much is shown, we inquire whether the defendant's activities give rise to piracy concerns, remembering that the pejorative

---

193 See supra text accompanying notes 185-92.
label in this case has a precise meaning: did the defendant's activities create market advantages not available to persons who reverse engineer in other technological fields? If not, we should find no infringement. Otherwise, we are protecting software technology more than other technologies without a policy basis—in other words, without knowing why we are doing it.

When the defendant's program competes directly with the plaintiff's original, proof of slavish copying of errors and traps can take a significance beyond simple proof of copying. If a competent engineer is honestly reverse engineering, he will have to examine each line of code as well as the general structure. If for no other reason, such examination improves efficiency. No competent engineer who engages in this process will copy nonfunctional or erroneous code, at least not very often. Consequently, proof of slavish copying should permit an inference of unfair market advantage secured by decreasing the time it takes to honestly reverse engineer the program. The defendant should be able to rebut this presumption through a paper trail demonstrating significant independent development effort and that the slavishly copied errors or traps do not evidence slavish copying of substantial portions of the operative code. Here, the defendant will have the burden of proof. On the other hand, in the absence of evidence of slavish copying, a general paper trail showing that the defendant thought about each stage of production and spent time and effort in the process typical of program production should be enough to rebut the plaintiff's prima facie case, notwithstanding substantial similarity of structure, for example.

When the defendant's program does not compete with the plaintiff's original, as in the program conversion cases like Whelan and SAS Institute, the piracy problem vanishes. Here the social interest in rapid and efficient expansion of computer software technology to all corners of society, as well as efficient competition among programmers, argues that we should encourage, rather than punish, such activity. Evidence of slavish copying in these cases is irrelevant to the underlying policy issues and should therefore be ignored.

3. *Relationship of Scope of Protection to Statutory Language*

The fundamentally different nature of computer programs in comparison with traditional copyrightable works has naturally led to a variety of judicial approaches to program protection as courts make incomplete analogy to one aspect or another of programs in applying traditional authorities and interpretations of the statute. Some of these approaches have involved technical, often literal, application of the statutory language. The formal treatment of programs under the Copyright Act as literary works, for example, has led many courts to

---

194This is not to say that the plaintiff in SAS Institute should not have had a claim for breach of contract. Here we are considering only the copyright issues.

195See supra text accompanying notes 82, 86, & 163-65. It also vanishes in a case like Strohorn, where a new videogame was created using a large part of the code for an existing game but the new game appealed to a different market from the old. See supra text accompanying notes 94-103.
invoke as authority cases involving novels and plays, as discussed above. 196 Two other literal applications of statutory language that lead to incorrect results when viewed in terms of the policies advocated herein have been adopted by some courts, with the support of some of the commentators. They are, first, treatment of a subsequent programmer's product as a derivative work, and second, treatment of a human-readable copy of a program made for the purpose of study as an infringement.

A. PROGRAMS AS DERIVATIVE WORKS

The current Copyright Act defines a derivative work as follows:

A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted... 197

This is potentially a very broad definition, especially if the examples listed are not regarded as limiting. At the extreme, any author who uses an existing work as a starting point can be accused of creating a derivative work, in the sense that the subsequent work is "based upon" the original.

Two commentators have advocated essentially this interpretation for programs. 198 Moreover, at least one court has held that the conversion of a statistical analysis program to run on a different type of computer was "substantially and pervasively" based upon the plaintiff's original, leading to the conclusion that the defendant's program was a derivative work. 199 Another court stated that similarities between programs would be "more significant" when considered as derivative works under the statutory language, but ultimately concluded there was at most similarity of ideas and denied copyright infringement without addressing the derivative work issue directly. 200

Congress did not intend through the insertion of an explicit derivative work definition in the 1976 Copyright Act Amendments to radically change the prior law to protect ideas, concepts, or methods contained in a work. The express denial of protection for such elements in § 102(b) would otherwise be superfluous. Consequently, before the product of a subsequent author can infringe the original author's right to prepare derivative works, it must contain substantial similarity of copyrightable elements sufficient to constitute infringement. 201 The derivative work provisions are therefore more properly

196 See, for example, supra text accompanying notes 97, 110, 137, & 158.
198 Conley & Bryan, supra note 13, at 114; Davidson, supra note 79, at 1095-96.
201 Nimmer on Copyright § 3.01.
viewed as explicitly defining the scope of protection in derivative works legally made\textsuperscript{202} rather than as defining a new class of infringing works.

As a result, if the idea/expression distinction is maintained in the computer program context, a subsequent program that is found to have taken only ideas (presumably based on externality and efficiency considerations) should be neither a copy nor a derivative work, notwithstanding that the ideas taken were "based upon" the original program. Moreover, if for policy reasons we abandon the idea/expression distinction for programs as unsuited to their technological nature, as advocated above,\textsuperscript{203} we cannot literally follow isolated pieces of statutory language designed to define rights in works for which the idea/expression distinction has meaning. The policy basis for program protection is piracy, and this must be balanced against the policy favoring rapid and efficient diffusion of technology. Under this policy, we want software engineers to learn from others and make further, cumulative innovations, limited only to the extent necessary to maintain the advantages that the original innovator in technology has traditionally enjoyed simply by being first in the market with a commercial product. Because technology develops cumulatively, all technological innovations could be considered derivative works, because they are built on the existing "base" of technological knowledge. A literal interpretation of the derivative work definition would stifle program technology. Any decision to apply it literally, therefore, should be based on a clearly articulated policy basis that meets this objection and not merely on slavish reading of the statutory language.

B. COPYING PROGRAMS IN HUMAN-READABLE FORM

Another problem with the literal application of copyright notions to programs arises out of the concept of what constitutes a "copy." Again, the fact that programs are intended to be used rather than read should cause us to rethink the purpose of prohibiting copying. Blind application of the traditional notion of infringement by copying leads to serious conflicts with the underlying policy for protecting programs.

Most programs, whether in object or source code, are distributed in a form that is not directly readable by human beings. They are intended to operate machines and so come in a form that permits such operation most rapidly and conveniently. The typical user has no desire to see the code, as her only goal is to use the program in her computer. If the technological innovations contained in the program are to be available to other programmers, however, they must be put in a form that the programmers can read and study. This may involve copying to another part of computer memory for analysis, or decompiling and printing line-for-line on paper. In either case, the question is whether such activities constitute infringing "reproduction" of the original program.\textsuperscript{204}

\textsuperscript{202}Protection in a derivative work extends, and is limited, to the material contributed by the author. 17 U.S.C.A. § 103(b).
\textsuperscript{203}See supra text accompanying notes 82-84 & 178 and text following note 192.
\textsuperscript{204}See generally Karjala, supra note 123, 8 Mich. YB. of INT'L L., at ______; 8 EUR. INTELL. PROP. REV. at 109, see also Laurie & Everett, Protection of Trade Secrets in Object Form Soft-
One’s initial response to the question above is that the answer is obviously affirmative, but that is because we are used to thinking of copyright in application to traditional works, all of which are intended for a human audience. Copying these works in a form that humans can comprehend is precisely the activity the statute has always aimed at preventing. Programs, however, are intended to be used in computers. If there were a way to guarantee that any printout of a program in human-readable form would not later be placed in a computer for use, the original programmer could not be legitimately concerned. The only value in putting a program in human-readable form (once later placement into another computer is prevented) is that it can be studied for use in creating other programs. When the value of this type of study is accepted in the policy analysis, any desire of the original programmer to impede the free use of his ideas cannot be deemed legitimate. He simply wants more protection for his efforts than we have ever given to technological innovation in other areas, and no one has offered a plausible basis for suddenly deviating from this policy in the case of computer program technology.

At least one court has recognized that dumping and analyzing program code does not establish piracy, partly because testimony from both sides established that this is standard industry practice. Another court, however, concluded that copying for the purpose of analysis—indeed for any unauthorized use whatsoever—constituted infringement. Such a holding, notwithstanding its basis in literal statutory language, constitutes a radical change in the protectionist direction. It means that programs distributed in non-human-readable form are not only protected against piracy but also against any taking of the technological ideas and methods that can be learned by human inspection of the work, which have heretofore always been freely available for use by others.

The facts of Hubco Data Products Corp. v. Management Assistance Inc.
raise this issue in a different and interesting way. The plaintiff was attempting to price discriminate among customers for its operating system by placing "governors" on less expensive versions that restricted memory and the capacity to make use of peripheral devices. The defendant offered to the plaintiff's customers a service, and later a computer program, that would remove the governors from the less expensive models and thereby convert them into more powerful operating systems. The system involved line-by-line comparison of the governed and ungoverned versions to determine where the governors were, after which it was a simple matter to remove them. However, effecting the line-by-line comparison required making a copy of the plaintiff's program.\(^{299}\) The court concluded that this constituted infringement.

Whether or not the defendant's activity in *Hubco* should be permitted is something over which reasonable people may differ. It seems clear, however, that neither traditional copyright policy nor the policy underlying protection of computer programs has anything to do with the problem. With traditional works we never had to inquire into the reason for copying, because ultimate human communication was always the goal. With programs, however, use for the function accomplished by the program is the goal, and prevention of copying for such use is the purpose of the protection against copying. If an automobile manufacturer placed mechanical governors in the car that limited its speed or otherwise restricted performance, copyright law would have nothing to say about the propriety of removing them. The plaintiff in *Hubco* wanted to do the same thing with its program, and copyright now gets into the act only because the device in question can be placed into a one-to-one correspondence with human-comprehensible symbols. It seems that the defendant's activities in a case like this, if they should be regulated at all, should be regulated directly, rather than indirectly through technical interpretation of a statute that was never designed for and has never previously been considered applicable to the problem.

V. THE INFLUENCE OF THE NEW PROTECTIONISM ON TRADITIONAL COPYRIGHT LAW

While some of the computer software cases have been more cautious than others, the above analysis shows the existence of a definite trend in favor of a

\(^{299}\) At first, the defendant made the copy by physical printout on paper. The program developed by the defendant to "degovern" the plaintiff's operating system involved entering the plaintiff's operating system into the customer's computer. The court ruled that both these activities constituted copying. A paper printout, reasoned the court, is "certainly a 'material object' in which the copyrighted object code is 'fixed,' " thus slavishly following the literal statutory language in the manner herein criticized. Moreover, the defendant's software package for breaking the governors included a copy of the plaintiff's ungoverned operating system. Again the court ruled that placing this into the customer's computer would be a copy, following the standard analysis that placing a work in a computer constitutes making a copy. *Id.* at 458.
much higher degree of protection for program technology than correct analogy to traditional copyright law would permit. Whether or not one favors this trend in the case of programs, we are already seeing an influence of the software-related cases on the level of protection offered to traditional copyrightable works, and again the direction is one of increased protection. Two cases, in particular, are almost shocking in their extension of copyright protection to works that could never have qualified before copyright got involved with protecting computer software. One case involves a method for playing the commodities market and the other a talking toy bear.

In Williams v. Arndt, 210 the plaintiff was a commodities trader and author of a booklet entitled Floor Trader’s Manual, which set forth a laboriously derived step-by-step method for trading various commodities. He sold the booklet to a limited number of his customers for prices ranging from two thousand to three thousand dollars. The defendant obtained a copy of the booklet and programmed the method in source code so that the numerous decisions required by the system to be made on the input data could be rapidly effected by computer. 211 The defendant began marketing the program, and the plaintiff sought an injunction and damages, asserting that the defendant’s source code was a translation of the plaintiff’s booklet.

Incredibly, the court agreed, without so much as a passing reference to § 102(b) of the Copyright Act. Although the defendant’s case seems to have been very poorly argued, 212 the court based its finding of infringement almost solely on the fact that the defendant’s program produced results identical to those calculated by hand following the method in the plaintiff’s booklet. 213 Obviously, any attempt to computerize the plaintiff’s system would do the same thing. If it did not generate the same results, it would not be the same system. Under § 102(b), the plaintiff’s copyright in the booklet does not extend to his system, but under the holding of this case, the plaintiff now has a life-plus-fifty-year monopoly on implementing his system by computer.

In Worlds of Wonder, Inc. v. Vector Intercontinental, Inc., 214 the plaintiff was the manufacturer and distributor of an animated toy bear named Teddy Ruxpin. A special cassette contained an audio track for songs and stories and a digital track for activating motors that controlled the movements of the bear’s eyes, nose, and mouth. The plaintiff created a series of cassettes with stories and songs. The defendant began to manufacture its own cassettes for use with the plaintiff’s bear, each with the audio and digital tracks required for compatibility with the bear but otherwise containing different stories.

\[^{210}626\text{ F. Supp. 571 (D. Mass. 1985).}\]
\[^{211}\text{The program apparently reduced to minutes the time it would take an experienced trader to reach a decision, as opposed to an hour following the booklet manually. Id. at 578.}\]
\[^{212}\text{According to the opinion, the defendant changed his story in midstream and tried to claim that he had developed the system independently. The judge understandably did not find this testimony credible. Id. at 579-80.}\]
\[^{213}\text{Id. at 579.}\]
\[^{214}\text{No. C86-2617, N.D. Ohio, Aug. 26, 1986.}\]
The court found a likelihood of success on the plaintiff’s claim that the defendant had infringed the plaintiff’s copyright in the bear as an audiovisual work, by comparing “the work created when [the plaintiff’s] cassette activated Teddy Ruxpin and the work created when [the defendant’s] tape activated Teddy Ruxpin.” Not surprisingly, the court found the “concept and feel” to have a general similarity. The court also found specific similarities in the tone, pitch, and pace of the narrator’s voice, the method used for signalling the end of a page, and the visual impression of the eye, nose, and mouth movement. The court never considered that the defendant was not selling a toy bear but rather a cassette. No case has ever held that duplicating the tone, pitch, and pace of a human voice in a sound recording is a copyright violation, and the identical visual impression of the bear’s movements necessarily derives from the bear’s mechanical construction. The case result is equivalent to saying that the creator of stories contained in a looseleaf binder with nonstandard holes can have a monopoly on further stories written on paper with holes that fit the binder.

These two cases, then, extend copyright protection to systems and to utilitarian objects, respectively. They reduce copyright to a vague but very general misappropriation statute. Future courts will probably recognize the very evident errors of *Arndt* and *Worlds of Wonder*, but the decisions still stand as striking examples of the protectionist trend in software protection and software-related cases.

**VI. CONCLUSION**

Despite their formal expression as combinations of symbols comprising literary works, computer programs are utilitarian articles created and developed for human use, not human comprehension. Although we have elected to protect computer programs under copyright law, this fundamental difference between programs and all other copyrightable works means that direct application of traditional copyright theories, policies, and precedents risks imposing on software technology artificial restrictions on free and efficient development. Unless firmly grounded on clear and concrete policy, the adjustments necessary for drawing the appropriate balance between incentive and the efficient diffusion of technology throughout society cannot properly be made. The reason is that traditional copyright notions like substantial similarity and the idea/expression distinction, and even technical concepts like the meaning of “literary work” or “derivative work” and what constitutes

---

215 *Id.* at 9.
216 In fact, independent fixation of sounds that imitate or simulate those in a copyrighted sound recording expressly do not violate the copyright holder’s exclusive right to make copies or derivative works. 17 U.S.C.A. § 114(b).
"reproduction," were developed when all protected works were aimed at a human audience. They do not fit well and may even make no sense at all when applied to works intended for use rather than human comprehension, unless guided by policy.

There is still room for debate on the policy bases for protecting the various types of software, but if in each case the policy basis for argument is clearly and concretely stated, its validity and plausibility can be more easily debated and separated from the conclusions derived from it. The judicial decisions in software-related cases show a strong protectionist trend, apparently based on an implicit policy view that creativity and investment in program production needs the incentive of a high level of protection. Some of the commentators have been more explicit in basing their arguments on such a policy. This policy basis is vague, however, and contains no countervailing policy indicating at what level protection should be limited or how the level of protection should vary with different types of works.

Moreover, creativity incentives alone do not distinguish software development from innovation in other technological fields, in which nonpatentable technology is left largely unprotected from commercial exploitation and adaptation except for the advantages derived from being first in the marketplace with a desirable product employing new concepts or combinations of technology. The reason for such a low level of legal protection in other technological fields is that the additional creative production perceived to follow from a higher degree of protection would not be sufficient to justify the resulting obstacles to the general diffusion and implementation of technological ideas. In other words, because technology develops cumulatively, protection might increase incentives at some early stage but would then prevent other workers from making further developments to carry the technology to successively higher stages. No one has offered a plausible basis for believing that the creation, development, or use of software technology is any different from other forms of technology in this respect. Unless and until such a basis is presented, the simple call for more incentives for software creation does not provide an adequate foundation for estimating the optimal level of protection for software. It also fails to provide a basis for addressing the question of whether the full panoply of copyright protection, including the seventy-five year period of protection and the absence of general compulsory licensing, remains appropriate after the levels of protection have been significantly increased.

This article puts forth piracy as the policy basis for distinguishing software from other forms of technology and affording software a different scheme of legal protection. Piracy is defined concretely as copying activity that substantially eliminates the competitive commercial advantages enjoyed by technological innovators in other fields who are first in the marketplace with a new product. When this definition is coupled with the technological
nature of programs as utilitarian works, the overly protective approach of the courts in software-related cases becomes evident and a framework for protecting software under copyright law that meets the antipiracy policy goal while remaining consonant with traditional levels of protection for technology comes into focus.