Campbell Law Review

Volume 14
Issue 1 *Winter 1991*

Article 1

January 1991

Copyright Law: Integrating Successive Filtering into the Bifurcated Substantial Similarity Inquiry in Software Copyright Infringement Cases: A Standard for Determining the Scope of Copyright Protection for Non-Literal Elements of Computer Programs

Maury M. Tepper III

Campbell Law Review

Volume 14 Winter, 1991 Number 1

ARTICLES

COPYRIGHT LAW: INTEGRATING SUCCESSIVE FILTERING INTO THE BIFURCATED SUBSTANTIAL SIMILARITY INQUIRY IN SOFTWARE COPYRIGHT INFRINGEMENT CASES: A STANDARD FOR DETERMINING THE SCOPE OF COPYRIGHT PROTECTION FOR NON-LITERAL ELEMENTS OF COMPUTER PROGRAMS

MAURY M. TEPPER, III*

ABSTRACT: The non-literal elements of a computer program, such as its user interface, are crucial in determining that program's success on the commercial market. Such non-literal elements represent a substantial portion of the development costs of a program, but they are quite inexpensive to copy. Courts are

* Attorney, Womble Carlyle Sandrige & Rice. Mr. Tepper practices in the firms Raleigh, North Carolina office, where he concentrates in intellectual property and computer law.

The author would like to thank Professor C. Edward Fletcher, III of the University of Cincinnati College of Law for his guidance, assistance and support in the creation and publication of this article.

An abbreviated version of this article has been entered in the Nathan Burkan Memorial competition in Copyright Law, Sponsored by ASCAP.

currently unable to agree on the extent to which copyright law offers protection to the non-literal elements of computer programs, leaving the industry uncertain and hesitant to develop new user interfaces. This article develops a principled approach for determining the proper scope of copyright protection for the non-literal elements of computer programs.

TABLE OF CONTENTS

In today's society, computers have become a part of the everyday lives of a large portion of the population, many members of which have no formal training in computer science.[1] The market for computer programs is so pervasive that unauthorized copying alone costs the software industry nearly $2 billion a year in lost revenue.[2] A number of mass-marketed programs, such as Lotus 1-2-3 and Wordperfect, have become so well known that there is even a market for "clones," programs that perform the same tasks as these well-known programs in a similar manner, but at a lower cost.[3] Because the market for "user friendly" programs is so well-developed, it is somewhat surprising that the scope of copyright protection for such programs is unsettled.

A specific instance concerns protection for "non-literal" elements of programs. Courts have been unable to agree on the extent to which a program's copyright protects the non-literal elements of that program. Non-literal program elements are those which are not manifested in the literal lines of the "code" of a computer program. The non-literal elements of a program include its structure, sequence, and organization; its screen displays; and its "look and

1. *See* Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1052 (1989) (computers have entered lives of many with no training in computer science).

2. Garfinkel, *Programs to the People*, TECHNOLOGY REV., Feb./Mar., 1991, at 53.

3. In Digital Communications Assoc. v. Softklone Distrib. Corp., the court dealt with a "clone" designed to perform the same function as the plaintiff's program "Crosstalk," which enabled computers to communicate with each other. 659 F. Supp. 449, 452-53 (N.D. Ga. 1987).

feel."[4] These non-literal elements are perceived by the user when the program runs; they cannot be perceived by merely reading the "text" of the program code. The user interface of a computer program is the portion of the program, made up principally of its screen displays and menus, that communicates information to the user and instructs the user how to proceed. The user interface, which is basically the only part of the program that the user perceives, is made up of a combination of non-literal elements.[5]

In 1980, following the recommendations of the National Commission on New Technological Uses of Copyrighted Works (CONTU), Congress amended the Copyright Act of 1976 (the Act) to make it clear that copyright protection is available for computer programs.[6] The first wave of cases brought subsequent to the 1980 amendments involved literal, or word-for-word, copying of a competitor's program.[7] Courts had little trouble determining that the

---

4. *See* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1239 (3d Cir. 1986) (non-literal elements of program are structure, sequence, and organization), *cert. denied*, 479 U.S. 1031 (1987); Broderbund Software v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (user interface is non-literal element). Although terms such as "structure, sequence, and organization" or "look and feel" may be colorful, at least some experts find them to be unhelpful in determining which elements of a program should be protected. For a discussion of the shortcomings of these titles, *see LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 20-21 (1989) [hereinafter *LaST Report*].

5. *See* Manufacturers Technologies v. Cams, Inc., 706 F. Supp. 984, 993 (D. Conn. 1989) (user interface often described as "look and feel" of program); Pilarski, *User Interfaces and the Idea-Expression Dichotomy, or Are the Copyright Laws User Friendly?*, 15 AIPLA Q.J. 325, 327 (1987). This Article will use the terms "look and feel;" "structure, sequence, and organization;" "user interface;" and "screen displays" individually and collectively to refer to a program's non-literal elements, because they are interrelated and because no clear definition separates them. For example, it is impossible to entirely separate the "look and feel" of a program from its "structure, sequence, and organization;" likewise, it is impossible to speak of "look and feel" independently of a program's "user interface," or vice versa.

6. Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3028. The Copyright Act of 1976 is located at 17 U.S.C. § 101 (1988). The 1980 amendments clarified the Act's applicability to computer programs by adding a definition of "computer program" and creating a new section giving users of programs the right to create backup copies and to adapt the programs in order to make them functional on the user's computer. *See id.* §§ 101, 117.

7. Menell, *supra* note 1, at 1048; Clapes, Lynch, and Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493, 1052 (1987) [hereinafter *Binary*

Act protected against such direct copying.[8] Beginning in the mid-1980's, however, a second wave of cases began to be brought, alleging infringement of the non-literal elements of programs.[9] Courts have had difficulty determining to what extent, if any, copyright law should protect these elements of computer programs. Because a program's non-literal elements represent a substantial portion of its development costs and play an important role in determining its marketability, it is important to develop a principled means of defining the scope of copyright protection in this area.[10] The purpose of this Article is to develop such a method.

In June of 1990, the United States District Court for the District of Massachusetts announced that the copyright of the spreadsheet program Lotus 1-2-3 was infringed by a competitor's use of the program's user interface.[11] The *Lotus* opinion emphasized the competitive nature of the software industry and brought into focus the lack of judicial agreement over just what elements of a program, beyond the program's literal code, are protected by the program's copyright.[12] Perhaps the uncertainty in this area is due to the fact that those who make law and policy do not understand the technical elements and workings of computer programs.[13] Perhaps

---

Bards].

8. *Binary Bards, supra* note 7, at 1052.

9. *Id.* Non-literal elements can be copied or emulated without directly copying a program's code. Such elements include a program's screen displays, its user interface, its look and feel, and its structure. *See id.*

10. *See* Note, *A Thousand Clones: the Scope of Copyright Protection in the "Look and Feel" of Computer Programs- Digital Communications Associates, Inc. v. Softklone Distributing Corp.*, 659 F. Supp. 449 (N.D. Ga. 1987)., 63 WASH. L. REV. 195 (1988) [hereinafter *Clones*] (user interface is single most important factor in marketability of computer program); Curtis, *Engineering Computer "Look and Feel": User Interface Technology and Human Factors Engineering*, 30 JURIMETRICS J. 51, 52 (1989) (user interface is crucial to marketing program); *id.* at 56 (over 40 percent of instructions in program are written to implement user interface).

11. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990). A program's user interface is made up of its screen displays and menus. The interface is how the program communicates with the user. *See supra* text accompanying note 5.

12. *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986) (copyright extends beyond program's literal elements to protect structure, sequence, and organization), *cert. denied*, 479 U.S. 1031 (1987); Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003 (N.D. Tex. 1978) (copyright does not protect input formats).

13. *Binary Bards, supra* note 7, at 1499.

it is due to the fact that it was impossible for the drafters of the Copyright Act to foresee the difficult issues created by rapidly advancing technology.[14] Whatever the cause, it is important to clarify the extent of protection offered by copyright law to non-literal elements of a computer program in order to end uncertainty in the software industry about a significant aspect of programming.[15]

This Article discusses the need for a principled means of determining the scope of copyright protection for the non-literal elements of a computer program. Part I begins by surveying the other forms of legal protection available for programs, demonstrating why each is less desirable than copyright. A second Part then provides a brief explanation of computer programs and the programming process. Part III presents basic copyright principles and doctrines, in order to frame a discussion of the issues involved in determining the scope of copyright protection for non-literal program elements. In Part IV, this Article surveys the major judicial decisions addressing this issue, identifying some of the problems that courts have encountered with it. Part V discusses the need for protection of non-literal program elements and refutes the principal arguments against offering such protection. In Part VI, this Article reviews the successive filtering test, a recent attempt to define the scope of copyright protection for computer software, explaining why this test alone is inadequate. Finally, the Article concludes with a proposal for the adoption of a bifurcated substantial similarity test that integrates the successive filtering approach for software infringement cases.

---

14. CONTU, for example, was aware of potential future difficulties: "Most infringements, at least in the immediate future, are likely to involve simple copying. In the event future technology . . . permits future infringers to use an author's program without copying, difficult questions will arise." NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 22 (1978), *reprinted in* 3 COMPUTER/LAW J. 53, 72 (1981) [hereinafter CONTU Report].

15. *See* Note, *Copyright Protection for Computer Screen Displays*, 72 MINN. L. REV. 1123, 1124 (1988) [hereinafter *Screen Displays*] (until uncertainty is resolved, software firms cannot be confident in ability to protect developments); Nimmer, Bernacchi, and Frischling, *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625, 630 (1988) [hereinafter *Structured Approach*] (uncertainty created by ad hoc nature of current computer copyright decisions hampers development in software field). For a discussion of the importance of user interfaces to the marketability of computer programs, see *supra* note 10.

I. THE UNSUITABILITY OF OTHER FORMS OF LEGAL PROTECTION

Although it does have some disadvantages, copyright law has emerged as the primary source of protection for computer programs.[16] In 1978, CONTU surveyed the available forms of legal protection for software and concluded that copyright protection was the most suitable.[17] Although copyright law is the primary source of protection for software, it is not the only available source. The following discussion briefly examines other available forms of legal protection for computer programs, outlining some of their disadvantages, in order to demonstrate that copyright doctrines should continue to be employed in determining the scope of legal protection for software.

A. Patent Law

In the most doctrinally pure sense, patent law would be the appropriate form of legal protection for computer programs.[18] Patent protection is available for "any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof."[19] Although certain fundamental processes, such as mathematical algorithms, have been excluded from patent protection on the grounds that they are too important to be subjected to private control, programs using such algorithms have been held to be patentable, as long as they do not preempt or merely recite the algorithms.[20] The Patent and Trademark Office (PTO) has implemented procedures for reviewing applications for patents on software programs and has issued a number of these patents.[21] Additionally, design patents, which last for fourteen years, are available for ornamental aspects of a program, such as

---

16. For a discussion of some of these disadvantages, see Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35, 47-50 (1989).

17. CONTU Report, *supra* note 14, at 16-18.

18. *See* Note, *Idea, Process, of Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 893 (1990) [hereinafter *Process or Expression?*] (patent law provides traditional mode of protection for utilitarian works).

19. 35 U.S.C. § 101 (1988).

20. Diamond v. Diehr, 450 U.S. 175 (1981); *see* Einhorn, *Copyright and Patent Protection for Computer Software: Are They Mutually Exclusive?*, 30 IDEA 265, 269 n.19 (1990).

21. For a listing of some of the software patents issued by the PTO, see Menell, *supra* note 1, at 1076.

certain elements of the user interface.[22] Design patents protect any "new, original and ornamental design for an article of manufacture."[23] On May 10, 1988, the PTO issued the first group of design patents on computer screen displays, which have been assigned to Xerox.[24] These design patents cover, among other things, design icons for a wastebasket, a file divider, a broken document, and a telephone.[25] Recent action by A.T.& T. to enforce its patent on software that allows a computer display to run several programs simultaneously may generate increased interest in the use of patents to protect computer programs.[26]

Although a computer program does seem to fit squarely within the requirements for a patent and a few companies have managed to obtain patent protection for their software, several characteristics of patent protection make it less suitable than copyright protection for programs. First, in order to be patentable, the design or process must not be "obvious."[27] Most programs, which basically do no more than describe and implement a method of performing a task, probably would not satisfy the non-obviousness requirement to merit patent protection.[28] At least one commentator has gone so far as to claim that less than one percent of all computer programs are patentable.[29]

Even if a program were able to satisfy the non-obviousness requirement of the patent statutes, the costs of obtaining a patent, in terms of both time and money, may prove to be prohibitive to a

---

22. Menell, *supra* note 1, at 1091.

23. 35 U.S.C. § 171 (1988). An "ornamental design" is one not dictated primarily by functional considerations. Power Controls Corp. v. Hybrinetics, Inc., 806 F.2d 234, 238-39 (Fed. Cir. 1986); *see also* Menell, *supra* note 1, at 1091.

24. Einhorn, *supra* note 20, at 269.

25. *See id.*

26. *See Patent Action on Software by A.T.& T.*, N.Y. Times, Feb. 26, 1991, at C1, col. 6. A.T.& T.'s action also demonstrates the danger of granting broad patent protection to computer programs. A.T.& T.'s patent will allow it to preclude others from using any form of its process, which has become "vital" to computer work stations, the fastest-growing part of the computer industry. Such action will likely drive some smaller companies out of the market, because A.T.& T. will be able to set the license fee for use of its process at any amount it wishes. *See id.*

27. 35 U.S.C. § 103 (1988).

28. *See* Menell, *supra* note 1, at 1076 (most programs do not satisfy non-obviousness requirement).

29. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 23 JURIMETRICS J. 339, 357 (1983).

programmer.[30] Before any patent may be issued, the PTO must conduct an examination of the application; the examination can be a very costly and time-consuming process.[31] Furthermore, as part of the patent application process, the applicant must disclose the art to the PTO.[32] This disclosure would allow public access to a program's process that could destroy any claim to trade secrecy protection.[33]

Aside from the difficulties in obtaining patent protection, several characteristics about the nature of that protection make it a less desirable choice, from a systemic point of view, than copyright protection. Patent protection is much broader than copyright protection, which only gives the holder the right to prevent others from making substantially similar copies of the protected expression.[34] Patent protection, in contrast, allows the holder to prevent others from making, using, or selling any device that accomplishes the same task in the same manner as the patented art.[35] Unlike copyrights, patents protect against all similar programs or processes, even those that are independently created.[36] The greater scope of protection afforded by patent law could effectively preclude competition in many areas of the programming industry, which strives to create better ways to perform the same tasks. The shorter term of protection granted to patent holders than to copy-

---

30. Menell, *supra* note 1, at 1076 (costs of obtaining patent may not be worth effort for product with short life cycle); *Process or Expression?*, *supra* note 18, at 894 (patent obtained with greater difficulty than copyright).

31. 35 U.S.C. § 154 (1988). The time required for a patent examination makes this form of protection particularly unsuitable for computer programs, which have a limited useful life and therefore require quick protection that will allow them to maximize their time on the market. *See* Menell, *supra* note 1, at 1076.

32. 35 U.S.C. §§ 111-12, 114, 154 (1988).

33. Although a programmer may not mind sacrificing trade secrecy protection if he is assured of receiving a patent, the doubtful availability of patent protection for most programs makes this sacrifice a costly one. The requirements for trade secrecy protection and the problems created for it by public disclosure are discussed *infra* at notes 42-53 and accompanying text.

34. 3 M. NIMMER & D. NIMMER, NIMMER ON COPYRIGHT § 13.01[B] (1990) [hereinafter NIMMER].

35. Menell, *supra* note 1, at 1075.

36. *See* Einhorn, *supra* note 20, at 270. Independent creation is a defense to copyright infringement. *See* 3 NIMMER, *supra* note 34, § 12.11[D], at 12-92.7-9. Thus, patent protection for a computer program could grant a monopoly, because it would not permit other programmers to perform the same process, even if those programmers independently develop a program to do so.

right holders reflects Congress' implicit recognition that the right to prevent others from using a process is "a powerful right that should be granted only for relatively short periods and under carefully circumscribed conditions."[37]

There may be an obstacle to obtaining a copyright registration for an article protected by a design patent. Copyright regulations provide that a work will be denied registration if a design patent has already been issued on that work.[38] There appears to be no rationale for this limitation imposed by the Copyright Office.[39] Additionally, there is a substantial question as to whether this regulation is valid. In *Application of Yardley*,[40] the United States Court of Customs and Patent Appeals held that the PTO could not require applicants to elect between copyright and design patent protection by refusing all applications already registered with the Copyright Office. At the moment, however, the safest course of action for a party seeking protection under both copyright and design patent laws is to register the copyright first, because unlike the Copyright Office, the PTO has no regulation that denies registration to a work already registered under the other statute.[41]

B. *Trade Secret Law*

Trade secret law protects all confidential matter, regardless of whether it is an idea, expression, or compilation of information.[42] Trade secret law, however, has proven to be an unsatisfactory

---

37. *Process or Expression?*, *supra* note 18, at 895. Patent protection only endures for 17 years. 35 U.S.C. § 154 (1988). Copyright protection, however, lasts for the life of the author plus 50 years, or in the case of a work made for hire, 75 years. 17 U.S.C. § 302(a), (c) (1988).

38. The regulations read as follows: "The potential availability of protection under the design patent law will not affect the registrability of a pictorial, graphic, or sculptural work, but a copyright claim in a patented design or in the drawings or photographs in a patent application will not be registered after the patent has been issued." 37 C.F.R. § 202.10(a) (1990).

The regulations present the same obstacle for works protected by an ordinary patent. Section 202.10(b) provides that copyright registration will be denied to any work for which a patent has been issued. 37 C.F.R. § 202.10(b) (1990).

39. *See* 1 NIMMER, *supra* note 34, § 2.19; Einhorn, *supra* note 20, at 274.

40. 493 F.2d 1387 (C.C.P.A. 1974). For a discussion of this opinion and its implications for joint registrations, see Einhorn, *supra* note 20, at 275-78.

41. *Yardley* did away with the PTO rule to that effect. *See* Einhorn, *supra* note 20, at 277.

42. *See* Harris, *A Market-Oriented Approach to the Use of Trade Secret or Copyright Protection (Or Both?) for Software*, 25 JURIMETRICS J. 147, 149 (1985).

means of protecting computer programs for several reasons. One disadvantage is that trade secret protection is largely a matter left to state law, and it therefore lacks national uniformity.[43] The standards which must be met to qualify for trade secret protection vary from state to state. For example, many, but not all, states require some degree of novelty for trade secret protection.[44]

The major disadvantage of trade secret protection is that it only protects "secrets." Once information is disclosed to the public, in whatever form, trade secret protection is forever lost.[45] Mass-marketing a program without "disclosing" it to the public presents theoretical and legal difficulties. Although a number of licensing schemes have been devised to allow for mass-marketing without public disclosure, their effectiveness is questionable at best, and one would be well advised not to rely exclusively on trade secret law for protection when marketing a program.[46]

It is also unclear whether trade secret protection is available in combination with copyright protection. At least one court has stated that the use of a copyright notice with a general publication is antithetical to the secrecy requirement of trade secret law.[47] Filing for a copyright registration arguably destroys trade secret protection, because the Copyright Act requires the registrant to deposit the material with the Copyright Office.[48] It is possible that the act of depositing material for a copyright registration may constitute public disclosure; the act of filing a patent application clearly does, because of the Patent Act's disclosure requirements.[49]

---

43. *See id.* (trade secret protection available under state law).

44. *Id.*

45. *See id.*

46. It is simply impractical to create any kind of clearly enforceable licensing agreement between the programmer and the consumer forbidding the consumer from disclosing any of the program publicly. The most popular of these arrangements, the "shrink-wrap" license, illustrates these difficulties. Under this arrangement, the consumer is purportedly bound by the terms of the licensing agreement when he opens the package containing the program and its manuals. If the consumer finds any of the terms disagreeable, he is to return the program. The enforceability of such an arrangement is far from clear. For a discussion of this and other licensing arrangements aimed at preventing public disclosure, *see id.* at 154-56.

47. Technicon Medical Information Sys. Corp. v. Green Bay Packaging, Inc., 687 F.2d 1032 (7th Cir. 1982), *cert. denied*, 459 U.S. 1106 (1983).

48. 17 U.S.C. § 411-12 (1988).

49. *See* Harris, *supra* note 42, at 158-59. The disclosure requirements of a patent application are discussed *supra* text accompanying notes 32-33.

Another disadvantage to trade secret protection is that the ability to "reverse engineer" a product can destroy protection under trade secret law.[50] Unlike copyright law, which prevents a person from producing substantially similar copies of a work, trade secret law only protects confidential information, and to the extent that a product can be reverse engineered, its contents are no longer confidential.[51] Finally, there is a risk that the Copyright Act will preempt state trade secret laws. Section 301 of the Copyright Act of 1976 states that the Act preempts all state law remedies which are equivalent to the exclusive rights given to a copyright holder under the Act.[52] Although the purpose of this section was to preempt common law copyrights, it may also operate to preempt trade secret protection, if that protection is deemed to be "equivalent" to the rights granted by the Copyright Act.[53]

C. *Trademark Law*

Trademark law may provide some limited protection to computer programs, but its value to software is questionable. Trademark law provides protection for the manner in which a program is sold, not for the content of the program.[54] Protection under the Lanham Trademark Act[55] is limited to preventing consumer confusion as to the source or origin of the goods.[56] Thus, trademark protection would only allow the holder to prevent others from marketing goods in such a way that the public might be misled into mistaking them for the trademark holder's goods. Because many products in the software industry are intended to be lower-cost versions of well-known programs, the value of trademark protection is marginal. Companies marketing "clones" or "knock-offs" attempt clearly to identify their products as lower-cost alternatives, not to pass them off as the more expensive programs themselves.

---

50. Trade secret law only protects secrets, and any program elements that can be recreated through examination clearly are not "secret." *See* Harris, *supra* note 42, at 160.

51. *Id.*

52. 17 U.S.C. § 301 (1988).

53. Harris, *supra* note 42, at 162; *see also* Luccarelli, *The Supremacy of Federal Copyright Law Over State Trade Secret Law for Copyrightable Computer Programs Marked With a Copyright Notice*, 3 COMPUTER/LAW J. 19 (1981).

54. Harris, *supra* note 42, at 150.

55. Act of July 5, 1946, ch. 540, 60 Stat. 427, *codified at* 15 U.S.C. § 1051 (1988).

56. *Id.* § 1114(a).

Because it is to their benefit to distinguish their clones from more expensive programs, software competitors are unlikely to commit trademark infringement, and therefore the Lanham Act is likely to play a very small role in the protection of software rights.

II. A TECHNICAL INTRODUCTION TO COMPUTER PROGRAMS

A. *The Programming Process*

In order properly to understand the issues related to the protection of computer programs, one must first understand both the structure of a computer program and the process by which it is written. The following discussion provides a brief outline of the basic elements of a program and should in no way be considered an exhaustive treatment of an area on which volumes have been written.[57]

Section 101 of the Copyright Act of 1976 defines a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[58] This definition contemplates two basic kinds of programs: operating systems and application programs. An operating system is a set of instructions that manage the internal functions of the computer and are executed directly.[59] Application programs are programs written to perform specific tasks for the user.[60] Application programs are written in "programming languages," such as FORTRAN or BASIC, which are easier for the programmer to work in than the binary code that computers "read."[61] Because a programming language must be interpreted for a computer by an operating system, an application program, written in programming

---

57. *See generally* L. PETERS, SOFTWARE DESIGN: METHODS & TECHNIQUES (1981); B. SCHNEIDERMAN, DESIGNING THE USER INTERFACE: STRATEGIES FOR EFFECTIVE COMPUTER INTERACTION (1987); R. PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH (1982).

58. 17 U.S.C. § 101 (1988).

59. These programs are written in binary code that can be processed by the computer. *See* Menell, *supra* note 1, at 1051. (operating system directs flow of information among computer's working parts and acts as interface between hardware and application programs); *Process or Expression?, supra* note 18, at 868-69 (computers only capable of directly executing binary code); *Screen Displays, supra* note 15, at 1132 (operating system manages internal functions).

60. Menell, *supra* note 1, at 1051; *Screen Displays, supra* note 15, at 1132.

61. *Process or Expression?, supra* note 18, at 868-69. The operating system "translates" the commands of an application program into binary code for the computer. *Id.*

language, is used "indirectly" in the computer, within the meaning of section 101's definition.[62] Because almost all programs on the consumer market are application programs, this Article uses the term "computer program" to refer generally to application programs.

Although there are numerous ways to write a computer program, some generalizations can be made about the process most programmers use. A programmer follows five basic steps in writing a program: (1) task definition, (2) flowcharting, (3) coding, (4) debugging, and (5) documentation.[63] A programmer begins the process of writing by defining the task to be accomplished by the program.[64] In order to define the task properly, the programmer may have to study the problem in detail and learn about the preferences of the intended user.[65] Next, the computer programmer flowcharts, or graphs out, the way in which the task will be performed.[66] As part of this diagraming process, the programmer will often find it useful to break the task into smaller subtasks and instruct the computer to perform each subtask in a logical order.[67] The process of breaking a task down into smaller subtasks can occur at any number of levels, depending on the complexity of the programmer's primary task.[68] There is no single right way to break down a task, and each programmer may be expected to come up with a different flowchart, depending on his personal programming preferences and experiences.[69] The way in which the various sub-

62. *Id.* at 869.

63. *See* Menell, *supra* note 1, at 1051; *Process or Expression?*, *supra* note 18, at 870-72.

64. Menell, *supra* note 1, at 1051; *Process or Expression?*, *supra* note 18, at 870; *Screen Displays*, *supra* note 15, at 1131.

65. For example, in order to properly define the task for a program to manage small dental labs, one programmer found it necessary to visit several labs and interview people there in order to understand the flow and workings of such labs. Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1225-26 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

66. Menell, *supra* note 1, at 1051.

67. *Process or Expression?*, *supra* note 18, at 870.

68. This process of starting from the most general task and breaking it down into smaller subtasks is generally referred to as "top-down" design. *See generally Structured Approach*, *supra* note 15, at 637.

69. *Id.* at 870-71. There are, however, some basic limitations on the way in which tasks are broken down. Generally, programmers will try to structure the flow of a program to maximize efficiency, use the least amount of memory, and maximize the speed at which information is communicated to the user. *See* Menell, *supra* note 1, at 1052-53.

tasks relate to each other and the order in which they are performed can roughly be referred to as the "structure" of the program.[70] Once the task has been broken down into its component parts and flowcharted, the flowchart is then translated into "source code;" that is, it is coded into one of the programming languages.[71] In order to operate on a computer, the program must also be translated into "object code," the binary code that computers process; this step is generally accomplished by the computer's operating system.[72] After coding has been completed, two steps remain: debugging and documentation. Debugging consists of testing the workings of the program and correcting errors in it, and documentation involves the preparation of users manuals and other materials to instruct the user on the program's operation.[73]

A unique characteristic of the software industry is its tendency to progress in a "stepping stone" method. Developers of new programs often build onto existing programs in order to save time.[74] The developers of the Macintosh computer, for example, based much of their work on Xerox's Star workstation.[75] Nationwide computer "bulletin boards," networks through which users share programs, have developed.[76] One programmer has even developed a licensing system, dubbed "copyleft," which allows programmers to borrow software freely for use in other programs.[77]

---

70. *Process or Expression?, supra* note 18, at 871.

71. *See Screen Displays, supra* note 15, at 1131. For an explanation of programming languages, *see supra,* notes 61-62 and accompanying text.

72. *Id.* at 1131-32 & n.36. Operating systems are discussed *supra* note 59 and accompanying text.

73. Menell, *supra* note 1, at 1051. Although both of these steps are important, they have little bearing on determining the scope of protection that should be afforded to programs. For a more complete treatment of these two programming steps, *see* D. BENDER, COMPUTER LAW - SOFTWARE PROTECTION § 2.06[3][e] (1988); Whelan Assocs. v. Jaslow Dental Laboratory, 797 F.2d 1222, 1231 n.21 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

74. *See* Menell, *supra* note 1, at 1057 (citing Karjala, *Copyright, Computer Software, and the New Protectionism,* 28 JURIMETRICS J. 33, 39-41 (1987)). One author states that "[t]he computer industry has long been characterized by the free sharing of programs and program parts among programmers." *Process or Expression?, supra* note 18, at 872.

75. *Taking the Stand: The Look-and-Feel Issue Examined,* PC MAG., May 26, 1987, at 155, 157. One of the founders of Lotus noted, "[n]obody does anything from scratch." Schwartz & Rosenberg, *Computing the Cost of Copyright,* NEWSWEEK, Aug. 27, 1990, at 52.

76. *See* 3 NIMMER, *supra* note 34, § 13.03[F][4], at 13-72.

77. Garfinkel, *supra* note 2, at 57.

B. *Limitations on Programming Choices*

Although programmers could theoretically carry out a task in nearly infinitely many ways, some ways are better than others, and a number of practical considerations limit the choices available to programmers. Programmers will generally seek to write programs in ways that require the least possible amount of computer memory space, perform the task quickly, minimize the amount of time required to write, and allow for easy access by the user.[78] Because it will rarely be possible for a programmer to accomplish all of these functions, programmers attempt to strike a balance maximizing those factors most important to the task at hand.[79] Thus, although certain considerations limit the programmer's choices, the decision on how to strike the proper balance in achieving each goal ultimately reflects a creative choice by the programmer.

Because application programs are written largely for the consumer market, a primary concern limiting the programmer's choices is that the program must be easy to understand and use. Because the cost of learning a new software system can be significant, businesses are very interested in programs that can be learned quickly and with a high retention rate.[80] In fact, the user interface has been called the "single most important factor" in determining a program's marketability.[81] The role of the user interface in choosing a program design has become so important that a new science known as Computer-Human Interaction (CHI) has sprung up to study user responses.[82] As a result, the design of the "user interface" of a program, the part of the program that communicates with the user, represents a substantial portion of the development costs of a program.[83]

---

78. *Process or Expression?*, *supra* note 18, at 869.

79. *Id.*

80. Curtis, *supra* note 10, at 53.

81. *Clones*, *supra* note 10, at 195 (citing *The Difficult Path to the Easy-to-Use Computer*, BUS. WK., Feb. 27, 1984, at 93).

82. *LaST Report*, *supra* note 4, at 26. This science, a branch of Human Factor Analysis, helps programmers find ways to minimize user learning time and the rate of user errors and to maximize performance speed. *Id. See also* Menell, *supra* note 1, at 1053.

83. *See Clones*, *supra* note 10, at 216 (user interface requires substantial investment of time and money). Over 40 percent of the instructions in a program are written for its user interface. Curtis, *supra* note 10, at 56. User interfaces are often created by a team of engineers, artists, and psychologists, in addition to programmers. *Screen Displays*, *supra* note 15, at 1132.

III. COPYRIGHT BASICS

A. *Requirements for Protection*

Copyright protection is based on the constitutional mandate "[t]o promote The Progress of Science and useful Arts, by securing for limited times to Authors and Inventors the exclusive Rights to their respective Writings and Discoveries."[84] Under this mandate, the aim of copyright law is not to grant absolute monopolies to creators, but rather to offer them protection only to the extent required to advance the public good by encouraging innovation.[85] Generally, copyright and patent laws strike a balance by granting limited monopolies to allow creators to recoup their development costs.[86] In an attempt further to refine this balance, Congress passed the Copyright Act of 1976, the current source of copyright protection.[87] Copyright protection under the Act gives the copyright holder a number of exclusive rights, including the right to reproduce the copyrighted work, the right to prepare derivative works based on the work, and the right to distribute copies of the work.[88] Section 117 of the Act contains a limitation on these exclusive rights specifically relating to computer programs. That section

---

84. U.S. CONST. art. I, § 8.

85. *See* Mazer v. Stein, 347 U.S. 201, 219 (1954) (reward to owner is secondary consideration of copyright law); United States v. Paramount Pictures, 334 U.S. 131, 158 (1948) (primary goal is promotion of public good).

The constitutional goal in establishing intellectual property protection reflects a view based on the expression of the great English jurist, Lord Mansfield:

> [W]e must take care to guard against two extremes equally prejudicial; the one, that men of ability, who have employed their time for the service of the community, may not be deprived of their just merits, and the reward for ingenuity and labour; the other, that the world may not be deprived of improvements, not the progress of the arts be retarded.

Sayre v. Moore, 102 Rev. Rep. 138 (1785), *quoted in* Damman, *Copyright of Computer Display Screens: Summary and Suggestions*, 9 COMPUTER/LAW J. 417, 419 n.3 (1989).

86. Without the chance to recover their development costs, authors and inventors would have no incentive to invest their resources into creating new processes and expressions. Granting more of a monopoly, however, could also retard progress by denying the public access to important works.

87. 17 U.S.C. § 101 (1988).

88. 17 U.S.C. § 106(1-3) (1988). Section 106 contains a more exhaustive list of the exclusive rights enjoyed by a copyright holder, but these three are the rights most relevant to computer programs. *See id.* § 106. A "derivative work" is defined in the Act as one "based upon one or more preexisting works, such as . . . abridgement, condensation, or any other form in which a work may be recast, transformed, or adapted." *Id.* § 101.

states that the owner of a copy of a program may copy that program for "backup" protection or adapt it in order to make the program useable on the user's computer system.[89] The exclusive rights granted by the Act endure for the life of the author (or authors) plus fifty years (or, in certain instances, for 75 years).[90] When any of the exclusive rights granted by the Act are violated, the copyright is infringed, and the copyright holder may bring an action in federal court to enforce his copyright.[91]

Section 102 of the Act lists seven categories of works of authorship which are copyrightable.[92] In 1964, the Copyright Office began to accept registrations for programs under the "literary works" category.[93] Courts have consistently held that programs are protectable under the Act as "literary works."[94] Additionally, the screen displays of a program seem to fit into the "audiovisual works" category of section 102, and, for a time, it was common practice to file separate registrations for the screen displays of programs under that category.[95] The Copyright Office announced a halt to that practice, however, in 1988, when it stated that it would only accept a single registration for each computer program.[96] Under this practice, one registration is deemed to extend to all copyrightable elements of a program, no matter which of the seven classes those elements fall into.[97]

In addition to fitting one of the seven classes of the Acts definition of a "work of authorship," a work must satisfy two other prerequisites in order to qualify for copyright protection. First, the

---

89. *Id.* § 117.

90. *Id.* § 302.

91. *Id.* § 501.

92. These classes are: "(1) literary works; (2) musical works, including any accompanying words; (3) dramatic works, including any accompanying music; (4) pantomimes and choreographic works; (5) pictorial, graphic, and sculptural works; (6) motion pictures, and other audiovisual works; and (7) sound recordings." *Id.* § 102(a).

93. Comment, *Manufacturers Technologies, Inc. v. Cams, Inc. - The Legal Fiction Created by a Single Copyright Registration of a Computer Program and its Display Screens*, 65 NOTRE DAME L. REV. 536, 542 (1990) (citing 11 BULL. COPYRIGHT SOC'Y 361, 362 (1964)).

94. For a discussion of some of these opinions, see *infra* text accompanying notes 133-288.

95. Comment, *supra* note 93, at 544.

96. 53 Fed. Reg. 21,817 (1988). The Copyright Office advised that a program should be registered under the class which predominates the work. *Id.* at 21,819.

97. *Id.* at 21,819.

work must be "original" in order to be copyrightable.[98] Originality does not rise to the level of the strict "novelty" requirement under the Patent Act.[99] In order to be original, a work simply must contain independent effort.[100] In addition to being "original," a work must be "fixed in [a] tangible medium of expression."[101] Courts have found that a program capable of being run in a computer is sufficiently "fixed" to qualify for copyright protection.[102]

B. Limits on Copyright Protection

1. The Idea/Expression Dichotomy

Because the purpose of the Act is not to grant absolute monopolies to authors, but rather to achieve the level of protection that most advances the public good, the Act and its judicial gloss contain some important limits on the scope of copyright protection. Perhaps the most important of these limits in the area of computer programming is what is commonly referred to as the "idea/expression dichotomy." Section 102 of the Act states that copyright protection extends only to expressions of ideas and in no way includes the ideas themselves.[103] It is clear from the statute that no idea can be copyrighted; to do so would be to give monopoly protection to ideas, which from common law times have been considered "as free as the air and as speech and the senses."[104] It is far from clear, however, just how to distinguish an unprotectable idea from the protected expression of that idea in a given case. Judge Learned Hand, the author of a number of important copyright decisions, noted that separating idea from expression was inevitably an ad hoc determination.[105] The lack of clarity in this area is somewhat disturbing when one reflects on the fact that just where the idea/expression line is drawn directly determines the

---

98. *See* 17 U.S.C. § 102(a) (1988) (Act protects original works of authorship).

99. For a discussion of the "novelty" requirement for patent protection, *see supra* notes 27-29 and accompanying text.

100. 1 NIMMER, *supra* note 34, § 2.01[B]. This standard requires only minimal contributions by the author. *See Screen Displays, supra* note 15, at 1126.

101. 17 U.S.C. § 102(a) (1988).

102. *See* 2 NIMMER, *supra* note 34, § 8.08.

103. 17 U.S.C. § 102 (1988); *see also* 3 Nimmer, *supra* note 34, § 13.03[A][1].

104. Desny v. Wilder, 46 Cal. 2d 715, 299 P.2d 257 (1956).

105. Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960).

scope of protection offered by the copyright laws.[106]

One of the best, and most frequently employed, tests for distinguishing idea from expression was set forth by Judge Hand in *Nichols v. Universal Pictures Corp.*[107] In that opinion, Judge Hand set forth what has come to be known as the "levels of abstraction test," writing:

> 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 all 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 expressions, his property is never extended.[108]

Under this test, a court begins by looking at the most specific levels of expression contained in a copyrighted work and moves to more and more general levels, until the expression becomes unprotectable ideas.[109]

2. *The Merger Doctrine*

Closely related to the statutory dichotomy between ideas and expression is the judicially-developed doctrine of merger. This doctrine holds that where the number of ways of expressing an idea is narrowly limited, copyright protection only protects against identi-

---

106. *See* Menell, *supra* note 1, at 1047.

107. 45 F.2d 119 (2d Cir. 1930), *cert. denied,* 282 U.S. 902 (1931).

108. *Id.* at 121. It is interesting to note that this process of moving from the specific to the abstract is the reverse of the process used by many programmers of starting from a general task and breaking it down into increasingly specific subtasks. For a discussion of this process, *see supra* notes 67-70 and accompanying text.

109. Note that Judge Hand refers to the *ideas* contained in a work. It is interesting that some courts addressing the protection of computer programs have felt constrained to define one idea underlying the program and separate the elements not essential to the idea as expression. This is contrary to Judge Hand's abstractions test, which recognizes that a work may contain numerous ideas. *See, e.g.,* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1236 (3d Cir. 1986) (purpose of program is its idea and everything not necessary to purpose is expression), *cert. denied,* 479 U.S. 1031 (1987); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (purpose of program is its idea and structure is expression).

cal copying of the particular expression.[110] Without such a limit, the first to copyright one of the few means of expressing a particular idea would have a virtual monopoly over that idea. Clearly, such a result would undermine the idea/expression dichotomy, and courts have therefore developed the merger doctrine to account for such situations. This doctrine has also been employed to deny copyright to an expression that is inseparable from the underlying idea, such that the expression has "merged" with the idea.[111] The purpose of the merger doctrine is the same as the purpose of the idea/expression dichotomy: it prevents monopoly control of ideas by denying protection to expressions that are inseparable from ideas.

The opinion in *Herbert Rosenthal Jewelry Corp. v. Kalpakian*[112] demonstrates this doctrine simply. In that case, the court refused to find copyright infringement by the defendant's jeweled bee pin, which was different from the plaintiff's only in the pattern of veins on the wings.[113] The court acknowledged that there was a great degree of similarity between the two pins, but it stated that the similarity was no greater than was inevitable from the idea of a jeweled bee pin.[114] The merger doctrine recognizes that in instances where the available means of expression are limited, such as in the *Rosenthal* case, it is necessary to draw the line for infringement closer to literal copying in order to avoid granting a virtual monopoly over an idea to the first to copyright a particular expression of that idea.

3. *Functionality*

Courts also have chosen to give only limited protection to utilitarian, or functional, works. Professor Nimmer notes that this is because the value of a utilitarian work is in its usefulness, rather than in the information it communicates.[115] This judicial doctrine

---

110. *See* Einhorn, *supra* note 20, at 271. In ordinary situations, copyright protection extends beyond identical copying and also forbids making substantially similar copies. *See* 3 NIMMER, *supra* note 34, § 13.01[B].

111. *See Process or Expression?, supra* note 18, at 877.

112. 446 F.2d 738 (9th Cir. 1971).

113. *Id.* at 740.

114. *Id.* at 742.

115. 1 NIMMER, *supra* note 34, § 2.18[A]-[D]. Therefore, giving broad copyright protection to a utilitarian work creates the danger of granting a monopoly over the useful process. *See Screen Displays, supra* note 15, at 1130.

has its roots in the seminal case of *Baker v. Selden*.[116] In that case, the United States Supreme Court held that the copyright for a book did not protect the accounting system discussed in the book or blank forms contained in the book for practicing the system, because the accounting system was a useful process.[117] Courts have used the *Baker v. Selden* rule to find blank forms, score cards, time planners, designs of clothing, and other functional articles uncopyrightable.[118]

Although utilitarian works are generally not protectable, any expression that can be separated from such works may be copyrighted. In *National Theme Productions v. Jerry B. Beck, Inc.*,[119] the court discussed the line between useful articles and aesthetic expression, there in the context of deciding the extent to which a series of children's animal costumes could be protected by copyright. The *National Theme* court began by acknowledging that costumes generally have "an intrinsic utilitarian function" that cannot be protected by copyright.[120] The court went on to state, however, that to the extent that they had features which could be identified separately and which were capable of standing alone as artistic expression, the plaintiff's costumes were copyrightable.[121] The court stated that it was necessary to draw a line to separate aesthetic from functional considerations in order to determine which elements could be protected.[122] Although utilitarian works are generally not copyrightable, the rationale of preventing monopoly control over a useful process does not apply to aesthetic, expressive elements of the article which can be conceptually separated from the process. Such elements are therefore copyrightable.

---

116. 101 U.S. 99 (1880).

117. *Id.* at 101.

118. *See, e.g.*, Whimsicality, Inc. v. Rubies Costume Co., 891 F.2d 452, 455 (2d Cir. 1989) (costumes not copyrightable); Cash Dividend Check Corp. v. Davis, 247 F.2d 458, 460 (9th Cir. 1957) (forms not copyrightable); National Theme Prods. v. Jerry B. Beck, Inc., 696 F. Supp. 1348, 1352 (S.D. Cal. 1988) (costumes not copyrightable); Januz Mktg. Communications v. Doubleday & Co., 569 F. Supp. 76, 80 (S.D.N.Y. 1982) (daily time planning chart not copyrightable).

119. 696 F. Supp. 1348 (S.D. Cal. 1988).

120. *Id.* at 1352.

121. *Id.*

122. *Id.* at 1353. This process is somewhat analogous to the process of drawing a line between the ideas and expression in a copyrighted work. For a discussion of that process, *see supra* notes 103-109 and accompanying text. *See also* Whimsicality, Inc. v. Rubies Costume Co., 891 F.2d 452 (2d Cir. 1989).

4. Scenes a Faire

Like merger, the scenes a faire doctrine is based on an understanding that in some instances, the available means of expressing an idea are limited, and in those instances, copyright protection should be similarly limited in order to avoid extending protection to the underlying idea. The scenes a faire doctrine applies when many of a work's elements are dictated by the work's subject or setting.[123] Under scenes a faire, stock characters or standard literary devices that have become inherent elements of a genre are denied copyright protection.[124] In a sense, the scenes a faire doctrine operates in the same way as merger, but it applies in situations where external conditions, rather than the nature of the idea itself, impose limits on the number of ways to express the idea.[125]

C. *Infringement Analysis*

Once the existence and scope of copyright protection have been established, a copyright holder can recover for infringement by showing copying. A copyright holder can show copying by either direct or circumstantial evidence. Because it is nearly always im-

---

123. For example, it would be difficult to write about the post-civil war reconstruction period in the American south without referring to the "stock" character of the carpetbagger. Judge Leon Yankwich, writing in 1951, explained the doctrine this way:

> These cases indicate that when you are dealing with a common idea, no matter how different the treatment may be, common elements will appear in both products. In so far as these common elements are distinct, they amount to creative originality. And, in so considering them, the similarities which are traceable to the common sources are disregarded. But similarities may appear which are inherent in a situation. The French refer to them as *scenes a faire*, - that is, scenes which *must* follow a certain situation.

Yankwich, *Originality in the Law of Intellectual Property*, 11 F.R.D. 457, 463 (1951).

124. *See* Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980); Q-Co. Indus. v. Hoffman, 625 F. Supp. 608, 616 (S.D.N.Y. 1985) (no unique expression in modules which would be inherent part of any prompting program); Plains Cotton Coop. Assoc. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256 (5th Cir.) (factors in program dictated by externalities of cotton market not copyrightable), *cert. denied*, 484 U.S. 821 (1987).

125. For example, nothing in the idea of the reconstruction period limits the choices of expressing that idea. A great amount of writing on the subject, however, has made the carpetbagger an indispensable element when discussing that period. *See supra* note 123.

possible for a plaintiff to show that a defendant literally transcribed the plaintiff's copyrighted work, courts have developed the substantial similarity test to allow plaintiffs to establish copying by circumstantial evidence.[126] Under this test, a plaintiff can create a presumption that copying occurred, thus shifting the burden of persuasion to the defendant, by showing that the defendant had access to the work and that the two works in question are substantially similar.[127]

Because access to a mass-marketed computer program is fairly easy to establish, infringement analyses in computer program cases generally turn on the substantial similarity inquiry. Although there is some confusion over the proper test for substantial similarity, courts examining literary works generally employ a bifurcated test, such as the test set forth by the United States Court of Appeals for the Ninth Circuit in *Sid & Marty Krofft Television Productions v. McDonald's Corp.*[128] Under this two-step test, the factfinder must first determine whether the underlying ideas of the two works resemble each other.[129] For this first inquiry, called the "extrinsic" branch of the test, the subject of the two works, materials used in the works, and other factors are examined using the assistance of expert testimony to dissect the two works.[130] If similarity of ideas is established by the extrinsic inquiry, the factfinder performs the "intrinsic" branch of the test by comparing the expression of the two works without the aid of expert testimony to determine from the standpoint of an ordinary observer whether the expressions in

---

126. *See* Comment, *supra* note 93, at 546; Comment, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test*, 68 MINN. L. REV. 1264, 1276 (1984) [hereinafter *Substantial Similarity*].

127. *See* Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 1000 (D. Conn. 1989); Digital Communications, Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 464-65 (N.D. Ga. 1987).

128. 562 F.2d 1157 (9th Cir. 1977). In that case, the plaintiffs alleged that the defendant's "McDonald land" commercials infringed its copyrighted series "H. R. Puffnstuff." *Id.*

129. *Id.* at 1164.

130. *Id.* The purpose of this "extrinsic" analysis is to determine whether the defendant used the plaintiff's work in preparing his similar work. In *Arnstein v. Porter*, a case using a similar bifurcated test, the court called on the factfinder, with expert assistance, to look at the two works as a whole, rather than dissecting the works, during the extrinsic inquiry. Arnstein v. Porter, 154 F.2d 464, 468 (2d Cir. 1946). Note that both tests compare the similarity of *unprotected* elements of the two works at this stage in order to make the initial determination of whether the defendant used the plaintiff's work.

the two works are substantially similar.[131] Although this test may appear at first blush to be somewhat complex, the thrust of it, and other similar tests, is to determine from the standpoint of an ordinary observer, whether the two expressions appear to be similar.[132]

IV. JUDICIAL ATTEMPTS TO DEFINE THE SCOPE OF COPYRIGHT PROTECTION FOR NON-LITERAL PROGRAM ELEMENTS

Courts applying the foregoing copyright doctrines to cases involving computer programs have come out with differing results. These courts have formulated various methods for determining which, if any, non-literal elements of a computer program merit copyright protection. The methods employed by these courts have developed no clear consensus as to the proper approach. A review of the reasoning of some of those cases will serve to illustrate some of the problems encountered by courts attempting to define the scope of copyright protection for non-literal elements of programs. This section analyzes the major decisions addressing this issue, discussing the advantages and disadvantages of the approach taken in each case.

What becomes apparent in examining these cases is that each court's analysis is flawed. The flaws differ from case to case, but the problems all derive from the failure of the courts to use the proper test in analyzing the appropriate level of copyright protection for non-literal program elements. Although several of the courts attempt to fashion such a test, none of them achieves one that is in harmony with the existing body of copyright law. The problems encountered by these courts would be solved best by the adoption of an improved mode of analysis described and discussed in Part VII below.

A. *Synercom*

In 1978, the United States District Court for the Northern District of Texas delivered one of the first opinions to discuss whether copyright protection should extend beyond a program's literal code in *Synercom Technology, Inc. v. University Comput-*

131. *Krofft*, 562 F.2d at 1164; *see also Arnstein*, 154 F.2d at 468. Note that for this inquiry, the factfinder only considers the expressions, or *protected* elements, of the two works.

132. *See Substantial Similarity*, *supra* note 126, at 1280. Because the ordinary observer is the focus of the intrinsic test, expert testimony and dissection of the works are not relevant. *Id.*

ing Co.[133] In *Synercom,* the plaintiff marketed a structural analysis program adapted from an early IBM program.[134] The strength of Synercom's program was that it was simplified for the user, largely through the development of instructional manuals and simplified user input formats.[135] Synercom sued a group of former employees who developed a competing program that accepted input in the same formats as Synercom's program, thereby allowing users to switch programs without any data re-entry.[136]

The *Synercom* court analyzed the copyrightability of the input formats in terms of the idea/expression dichotomy. Judge Higginbotham noted initially that the plaintiff's instruction manual was sufficiently original to be protected by copyright and that the defendants had copied the input formats from Synercom's copyrighted manual.[137] The question, then, was whether the input formats were part of the manual's expression, protected by its copyright, or were unprotectable ideas.[138] In order to demonstrate the nature of the formats, Judge Higginbotham analogized them to the "figure-H" pattern of an automobile gear-stick.[139] Although copyright would protect various expressions of the idea of a "figure-H" pattern, such as drawings in a driver's manual or prose descriptions of the pattern, it would not extend to the idea of the pattern itself, according to the court.[140] Thus, although the copyright owner could prevent others from copying his description or drawing, he could not prevent them from using the underlying idea of the pattern.[141] According to Judge Higginbotham, the ordering and sequencing of data (input formats) was like the "figure-H"

---

133. 462 F. Supp. 1003 (N.D. Tex. 1978).

134. *Id.* at 1005-06. For a discussion of how the practice of "building" new programs on existing programs is common in the software industry, *see supra* text accompanying notes 74-77.

135. *Id.* at 1006. The simplified input techniques of Synercom's program resulted in less training time for users and more efficient outputs. *Id.* at 1007.

136. *Id.* at 1008. Synercom had protected its input formats with a copyright notice. *Id.* at 1007. By designing its program to accept Synercom's input formats, the defendants apparently hoped to benefit from Synercom's investment of approximately $500,000 to train customers to use that format. *Id.* at 1008.

137. *Id.* at 1011. For a discussion of the originality requirement for copyright protection, *see supra* text accompanying notes 98-100.

138. *Id.* at 1011-12.

139. *Id.* at 1013.

140. *Id.*

141. *Id.*

pattern and was therefore an uncopyrightable idea.[142] The *Synercom* court held that expression could only be protected to the extent that it involved stylistic creativity beyond sequence and arrangement.[143] The court thus concluded that although the plaintiff's manuals were copyrightable, the formats were not.[144]

The *Synercom* opinion represents the lowest level of protection afforded by a court to non-literal elements of a program. Perhaps because the infringement claim was based on the copyright for the plaintiff's instruction manual, rather than the program, the court more or less limited that copyright's protection to preventing literal copying from the manual.[145] Although Judge Higginbotham's gear-stick analogy is intriguing, it is not helpful. The analogy, by giving an example of an idea and several expressions of it and then comparing the plaintiff's formats to the analogy's idea in order to conclude that the formats are not protectable, does no more than arrive at the proposition it assumed implicitly at its start: that the input formats were unprotectable ideas. The *Synercom* opinion denied the plaintiffs protection for formats which were developed at considerable cost, and which may have involved creative effort.[146]

B. E.F. Johnson

In 1985, the United States District Court for the District of Minnesota addressed the issue of protecting non-literal elements of a computer program by copyright in *E.F. Johnson Co. v. Uniden Corp. of America*.[147] In that case, the plaintiff sued for a preliminary injunction to prevent the defendant, Uniden, from marketing a program that would compete with the plaintiff's program for regulating radio communications systems.[148] In the process of developing its competing program, Uniden had "disassembled" the plaintiff's software and studied its structure.[149] When the plaintiff's engineers conducted a line-by-line analysis of Uniden's pro-

142. *Id.* at 1013.
143. *Id.* at 1014.
144. *Id.*
145. *See id.* at 1011.
146. The *Synercom* court did not consider whether other means of expressing the sequencing of data were available to the defendants, and so it is difficult to determine whether the input formats were in fact protectable expression.
147. 623 F. Supp. 1485 (D.C. Minn. 1985).
148. *Id.* at 1487.
149. *Id.* at 1490. Uniden used flowcharts to accomplish this task. *Id.*

gram, they concluded that copying had occurred.[150] The court evaluated the similarity of the two programs in order to determine whether infringement had occurred.[151] Under the traditional test for substantial similarity, the trier of fact is asked to determine whether the challenged work took enough material from the plaintiff's work so that an ordinary observer would recognize that there was borrowing.[152] The *Johnson* court noted, however, that application of the ordinary observer test to computer programs presented problems, because the average layman was unable to comprehend the elements of a program and their functions.[153] Because of these problems, the *Johnson* court adopted an "iterative" test for substantial similarity, which required a plaintiff to demonstrate that (1) the defendant used the plaintiff's work in preparing his copy and (2) the defendant's work was an "iterative" reproduction, or one based on exact reproductions of parts of the plaintiff's work.[154] The court went on to note that in the case before it, substantial similarity would be found under both the traditional and the "iterative" tests.[155]

After concluding that there was substantial similarity between the two programs and therefore copyright infringement, the court addressed the defendant's argument that many elements of the plaintiff's program were taken from pre-existing material, and therefore the program was not copyrightable.[156] The *Johnson* court found that although some elements of the plaintiff's program were taken from the public domain, the program possessed sufficient originality to be copyrightable.[157] Finally, the *Johnson* court rejected Uniden's claim that the plaintiff's program was an unpro-

---

150. *Id.* The most incriminating evidence of copying discovered by the engineers was that the disassembled code of Uniden's program bore the plaintiff's copyright notice. *Id.* at 1492.

151. *Id.* at 1492.

152. *Id.* at 1492-93. For a discussion of the bifurcated substantial similarity test adopted by the ninth circuit, *see supra* notes 128-132 and accompanying text.

153. *Id.* at 1493. For a proposal to avoid this problem, *see infra* text accompanying notes 350-360.

154. *Id.* at 1493. The *Johnson* court based its test on one suggested in *Substantial Similarity, supra* note 126.

155. *Johnson*, 623 F. Supp. at 1493.

156. *Id.* at 1498. For a discussion of the practice of using pre-existing material when designing a computer program, *see supra* notes 74-77 and accompanying text.

157. *Id.* at 1499. For a discussion of the originality requirement for copyright protection, *see supra* text accompanying notes 98-100.

tectable idea.[158] The court stated that if other programs could be written to perform the same function as the plaintiff's program, then the plaintiff's program was copyrightable expression.[159]

Although the *Johnson* court did offer protection to the non-literal structure of the plaintiff's program, its reasoning does not offer much hope for future protection of non-literal program elements. Under the court's "iterative" test for infringement, non-literal elements would only receive protection when they are infringed in conjunction with some literal copying.[160] The *Johnson* court chose to adopt the iterative test because it was concerned that juries would not be able to comprehend complex program codes. In effect, the iterative test limits the scope of copyright protection in a way similar to the merger doctrine, but in situations where the available means of expression are not necessarily limited.[161] Ease of application is simply not a sufficient justification for adopting a test that deprives a plaintiff of otherwise available protection.

Additionally, the *Johnson* court appears to have conducted its analysis in reverse order. The court determined that there was infringement before it addressed Uniden's argument that the plaintiff's program was an unprotectable idea.[162] Conducting an infringement analysis in this way creates a danger that juries may find infringement based on similarity of *unprotectable* ideas by comparing two works for substantial similarity before sorting out the unprotected elements.[163] The *Johnson* court should have first

---

158. *Id.* at 1502.

159. *Id.* Because a comparable program could have been written without copying, the court found infringement and granted a preliminary injunction against Uniden. *Id.* at 1502-03.

160. In other words, the plaintiffs in *Johnson* only received protection for the structure of their program because they had the good fortune of finding blocks of their program code copied by Uniden. *See id.* at 1492.

161. The merger doctrine, like the iterative test, draws the line for determining infringement close to literal copying. The merger doctrine, however, only draws this line when the choices of ways to express an idea are narrowly limited. The iterative test has no similar justification, other than the fact that it is easy to apply. For a discussion of the merger doctrine, *see supra* notes 110-114 and accompanying text.

162. *Johnson*, 623 F. Supp. 1493-99; *see supra* text accompanying notes 156-159.

163. Although the bifurcated substantial similarity test also takes similarity of ideas into account, it does so only in the extrinsic inquiry, to determine whether the plaintiff's work was used by the defendant. For the intrinsic analysis, the finder of fact compares only the protected elements of the two works for simi-

determined which elements of the plaintiff's program were protectable and then compared only those elements to determine whether infringement occurred.[164]

C. *Q-Co Industries*

In 1985, the United States District Court for the Southern District of New York expanded the potential for copyright protection of non-literal program elements in *Q-Co Industries v. Hoffman*.[165] The plaintiff in *Q-Co* sued for a preliminary injunction to prevent a group of former employees from marketing a teleprompter program modeled after the plaintiff's.[166] The *Q-Co* court noted that the only differences between the plaintiff's and defendants' programs were those that arose from hardware limitations in the different machines the programs ran on.[167] The court found that the structure of the two programs, and a few of their textual sections, were very similar, and it credited expert testimony that given the differences in the hardware for which the programs were designed, this similarity was highly unlikely, absent use of the plaintiff's program by the defendants.[168] The court concluded that although direct copying was impossible, because the defendants' program ran on a different machine and in a different language than the plaintiff's, the defendants did use the "structure and concept" of the plaintiff's program.[169]

According to the *Q-Co* court, the copying of the structure of the plaintiff's program did not amount to copyright infringement in that case. Referring to the idea/expression dichotomy, the court concluded without analysis that the copying of the plaintiff's four program modules was copying of ideas only.[170] The *Q-Co* court noted that there was no expert testimony establishing that the

---

larity. For a discussion of this test, see *supra* text accompanying notes 128-132.

164. For additional criticism of the iterative test, *see Structured Approach*, *supra* note 15, at 633-34.

165. 625 F. Supp. 608 (S.D.N.Y. 1985).

166. *Id.* at 613. The plaintiff's program ran on an Atari 800-XL computer. *Id.* at 610. The defendants developed a version of the program to run on IBM computers. *Id.* at 613.

167. *Id.* at 613. The IBM-PC that the defendants used had a more restricted screen display capacity than the Atari 800, and thus there were some differences in the screen displays of the two programs. *Id.*

168. *Id.* at 614.

169. *Id.* at 615.

170. *Id.*

plaintiff's modules contained any unique expression, and the court stated that the same modules would be an essential part of any tele-prompting program.[171]

Although it found no infringement, the *Q-Co* opinion is significant because it recognizes the possibility that there might be protectable expression in the structure of a program; there simply was no evidence of any in that case. Also, the court's recognition that if the program modules were an essential part of any program, they should not be protected by copyright is an implicit application of the scenes a faire doctrine, which denies copyright protection to devices that are standard to a given theme.[172] Because the *Q-Co* court was deciding a motion for a preliminary injunction, it did not have a full record before it. The *Q-Co* opinion, however, contains some reasoning which could have significant implications for the protection of non-literal elements of computer programs.

D. Whelan

The United States Court of Appeals for the Third Circuit extended broad protection to the structure of a program in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*[173] The defendant, Jaslow Dental Lab, had hired the plaintiff to write a program to take care of the business needs of a dental laboratory.[174] After several years of marketing the program for the plaintiff, Jaslow Lab decided that there was a market for a program that would run on smaller computers.[175] To meet this need, Jaslow developed a program in BASIC, which Jaslow advertised as a "new version" of the plaintiff's program.[176] The trial court found that Jaslow had infringed Whelan's copyright because the overall structure and organization of Jaslow's program was similar to that of the plaintiff's.[177]

---

171. *Id.* at 616.

172. For a discussion of the scenes a faire doctrine, *see supra* text accompanying notes 123-125.

173. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

174. *Id.* at 1225. The program was written for an IBM Series One computer. *Id.*

175. *Id.* at 1126. EDL, the language in which the plaintiff's program was written, apparently was not suited for the computers used by many smaller dental labs. *Id.*

176. *Id.* at 1226-27.

177. *Id.* at 1229. The court was assisted in this determination by expert testimony that most of the file structures and the screen outputs of the programs were

The *Whelan* court began its analysis with a discussion of how computer programs are written. After detailing the steps that a programmer goes through in writing a program, the *Whelan* court noted that the greatest expense in developing a program was attributable to the development of its structure and logic, and not to the actual "coding" into computer language.[178] Next, the court turned to an analysis of whether the district court properly found that Whelan's copyright was infringed, relying on the substantial similarity test.[179] The court noted that the traditional test for substantial similarity called for a bifurcated determination, the first relying on expert testimony, and the second relying on the jury's impression as ordinary observers of the similarity of the works in issue.[180] Because it found that computer programs were complex and indecipherable to the ordinary observer, the *Whelan* court rejected the traditional test for substantial similarity in favor of a single inquiry based on both lay and expert testimony.[181]

In order to apply the substantial similarity test, the *Whelan* court next had to determine the scope of copyright protection for the plaintiff's program. Starting from the premise that copyright clearly protected the "literal" code of a program, the court analyzed the Act to see what other elements of a program were protected by copyright.[182] The court noted that in other contexts, infringement had been based on findings of substantial similarity between two works, even absent literal copying.[183] The *Whelan* court saw no reason why computer programs should be distinguished from other works; thus, absent some limitation, copyright would protect the "total concept and feel" of a program to the

---

nearly identical. *Id.* at 1228.

178. *Id.* at 1230-31. For a discussion of the process of writing a program, *see supra* text accompanying notes 63-77.

179. *Id.* at 1232-33.

180. *Id.* at 1232. For a discussion of the bifurcated substantial similarity test, *see supra* text accompanying notes 128-132.

181. *Id.* at 1233. The court cited *Johnson* as an example of its new test. *Id.*

182. *Id.* at 1233-34.

183. *Id.* The court referred to cases which based findings of infringement on similarity of the "total concept and feel" of two works. *Id. See, e.g.,* Twentieth Century-Fox Film Corp. v. MCA, Inc., 715 F.2d 1327, 1329 (9th Cir. 1983) (plot similarities of two works may be basis for finding infringement); Sid & Marty Krofft Television Prods. v. McDonald's Corp., 562 F.2d 1157, 1167 (9th Cir. 1977) (similarities between characters in two works established by "total concept and feel" of two productions); Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) (copyright not limited to literal text).

same extent as any other literary work.[184]

One limitation on the extent of copyright protection is the idea/expression dichotomy.[185] The *Whelan* court conceded that it was difficult to separate idea from expression in any principled way, calling the distinction "elusive."[186] The *Whelan* court chose to define the idea of a program in terms of the end to be achieved by the work.[187] Therefore, according to the court, the idea of the program was its purpose, and everything not necessary to that purpose was copyrightable expression.[188] In that case, the court noted that the purpose of the program was to manage a dental laboratory, and because there were a number of different ways to structure a program in order to achieve that purpose, the structure of Whelan's program was protected expression.[189] The *Whelan* court noted that its formulation gave programmers the most incentive to develop new programs by protecting the structure of programs, which represented a substantial portion of programming efforts.[190]

The *Whelan* court acknowledged that its opinion appeared to be in conflict with Judge Higginbotham's decision in *Synercom*.[191] The *Whelan* court noted, however, that it was possible that the *Synercom* holding was based largely on Judge Higginbotham's conviction that the structure of the input formats in that case could not be separated from their underlying ideas.[192] The *Whelan* court cautioned, however, that to the extent that the *Synercom* decision held that there was a difference between the copyrightability of computer programs and other literary works, the *Whelan* court rejected that decision.[193]

The *Whelan* court correctly identified the need to strike the appropriate balance between protection and dissemination of in-

---

184. *Whelan*, 797 F.2d at 1234.

185. For an explanation of this limitation on copyright, *see supra* text accompanying notes 103-109.

186. *Whelan*, 797 F.2d at 1235.

187. *Id.* at 1236.

188. *Id.*

189. *Id.* at 1236 n.28.

190. *Id.* at 1237. The court had earlier established that the greatest expense in writing a program was attributable to developing its structure and logic. *See id.* at 1230-31.

191. *Id.* at 1239.

192. *Id.* at 1239 (discussing Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003 (N.D. Tex. 1978)).

193. *Id.* at 1240.

formation to promote progress and the public good.[194] The means adopted by *Whelan* in order to achieve that balance, however, have been widely criticized as overbroad.[195] The principal problem with the *Whelan* decision is the broad way it defines the single idea of a work as the work's purpose.[196] Under this rule, anything not essential to accomplishing the major purpose of a work is protectable expression; this determination fails to take into account limits that may be imposed on copyrightability by traditional copyright theories and by the practicalities of programming.[197]

Another potential problem is presented by the *Whelan* court's abandonment of the ordinary observer branch of the substantial similarity test. The *Whelan* court was correct in stating that laypersons are not qualified to assess the technical components of computer programs.[198] The court failed to note, however, that technical assessments are only required for the extrinsic branch of the substantial similarity test.[199] The thrust of the substantial similarity test remains whether an ordinary observer would conclude that the works are substantially similar.[200] The intrinsic inquiry, which is the heart of the substantial similarity test, only requires the examination of protected expression, and laypersons should be qualified to assess the similarities between such expression, which consists of user interfaces and other non-literal program elements

194. *See id.* at 1235. The constitutional basis of copyright law defines the purpose of protection as the promotion of progress. *See* U.S. CONST. art. I, § 8. For a discussion of the constitutional mandates for copyright law, *see supra* text accompanying notes 84-87.

195. *See, e.g., Structured Approach, supra* note 15, at 629-30 (although *Whelan* reached correct result its broad language and sweeping rule may extend protection too far); *Process or Expression?, supra* note 18, at 881 (single virtue of *Whelan* test is ease of application).

196. *See Whelan,* 797 F.2d at 1236. For a rejection of the concept that a work has only one idea, *see supra* note 109; *see also Computer Assocs. Int'l Inc. v. Altai Inc.*, 20 U.S.P.Q.2d 1641, 1650 (E.D.N.Y. 1991) (computer program made up of sub-programs and each sub-program has at least one idea).

197. The *Whelan* court acknowledged that some similarities in structure could arise in legitimate ways, but the court failed to devise a means of eliminating these similarities from the infringement analysis. *See Whelan,* 797 F.2d at 1248 n.47. For a discussion of some of the limits imposed by traditional copyright doctrines, *see supra* text accompanying notes 103-125.

198. *See Whelan,* 797 F.2d at 1233.

199. For a discussion of the bifurcated test for substantial similarity, *see supra* text accompanying notes 128-132.

200. *See supra* text accompanying note 132 for a discussion of this point.

that were designed to be understood by laypeople in the first place![201]

E. Broderbund

The United States District Court for the Northern District of California expanded the *Whelan* holding to apply to computer screen displays in *Broderbund Software, Inc. v. Unison World, Inc.*[202] The plaintiffs in *Broderbund* developed a print shop program which enabled users to create and print greeting cards, signs, and posters.[203] After the defendant, Unison World, unsuccessfully attempted to negotiate with Broderbund for the rights to create an IBM version of the Print Shop program, the defendant released a competing program which ran on IBM computers.[204] The *Broderbund* court began its analysis by discussing limitations on the scope of copyright protection, such as merger and the idea/expression dichotomy.[205] The court rejected Unison World's contention that the idea and expression had merged in that case, referring to an existing card-printing program designed for children, which had different menu screens, as evidence that there were alternative ways of expressing the idea in the plaintiff's program.[206] Additionally, the court rejected Unison World's suggestion that its choice of formats was limited by mechanical and utilitarian constraints, stating that the structure of Broderbund's program was based on aesthetic, rather than utilitarian, considerations.[207]

Turning to the question of copying, the *Broderbund* court found enough direct evidence of copying by Unison World to support a finding of infringement.[208] The court went on, however, to

---

201. For a discussion of the purpose of user interfaces, *see supra* text accompanying notes 80-83.

202. 648 F. Supp. 1127 (N.D. Cal. 1986).

203. *Id.* at 1129-30. The plaintiff's program ran on an APPLE computer. *Id.* at 1130.

204. *Id.* at 1130-31. During negotiations, Unison World's programmers attempted to faithfully re-create the print shop program on an IBM system, but after negotiations broke off, the programmers were permitted to make "enhancements" to the program. *Id.* at 1131. By this time, however, Unison World's programmers had already finished the menu screens and abut 10 other screens, which were left intact, in addition to the plaintiff's user interface. *Id.*

205. *Id.* at 1131-32.

206. *Id.* at 1132. The court followed the *Whelan* court's opinion, defining the idea of the plaintiff's program as its purpose: the creation of greeting cards, banners, and posters. *Id.* at 1133.

207. *Id.* at 1134.

208. *Id.* at 1135.

analyze the circumstantial evidence of copying under the substantial similarity test.[209] The *Broderbund* court acknowledged *Whelan*'s new integrated test, but it followed the traditional bifurcated test, which admits expert testimony to prove similarity of underlying ideas and requires the ordinary observer to assess whether there is substantial similarity between the expressions of those ideas.[210]

Broderbund shares *Whelan*'s recognition that the proper balance of protection must further the progress of science.[211] *Broderbund* also shares in the *Whelan* error of defining the single idea of a program as its purpose.[212] This broad definition of idea presents the danger of overbroad protection.[213] The *Broderbund* court has been criticized for misinterpreting the *Whelan* holding by applying it to computer screen displays, in addition to program structure.[214] Although *Broderbund* may have been incorrect in relying on *Whelan,* its conclusion that any aesthetic expression in a computer's screen displays can be protected by copyright appears to be correct, and it is certainly in line with other cases which protect the expressive elements in functional articles.[215] *Broderbund*'s

---

209. *Id.* at 1136.

210. *Id.*

211. *See id.* at 1134 (no danger in this case that copyright will create monopoly).

212. *See id.* at 1133. For a discussion of the *Whelan* reasoning, *see supra* text accompanying notes 185-190.

213. *See supra* notes 195-197 and accompanying text. The test employed by these two courts can be sharply contrasted with Judge Hand's abstractions test. Under Judge Hand's test, the factfinder begins with specific, comprehensible expressions and gradually eliminates the more general, unprotectable ideas from them. Under the *Whelan/Broderbund* test, however, a factfinder must begin by defining the sole idea underlying the work, and then term everything not essential to that idea as expression. For a discussion of the abstractions test, *see supra* text accompanying notes 107-109.

214. *See, e.g.,* Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 992-93 (D. Conn. 1989) (*Broderbund* appears to have misinterpreted *Whelan*); Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455 (N.D. Ga. 1987) (*Broderbund* represents overexpansive reading of *Whelan*). According to these courts and to commentators, *Whelan* held only that similarity of screen displays could serve as indirect evidence of copying, and *Broderbund* misread *Whelan* to state that screen displays, in addition to program structure, were protected by copyright. *See Clones, supra* note 10, at 201 n.52; *Screen Displays, supra* note 15, at 1145; Damman, *supra* note 85, at 425.

215. Although at the time, *Broderbund*'s holding may have been objectionable in light of the practice of filing separate registrations under the "audiovisual works" category to protect a program's screen displays, the position of the Copy-

rejection of the defendant's merger argument on the grounds that there was another program which printed cards in a different manner than the plaintiff's was somewhat inaccurate.[216] Under merger, the inquiry should have been whether the means of expression available were narrowly limited, not whether at least one other means of expression existed.[217] The *Broderbund* court took a great step forward by recognizing that mechanical and utilitarian constraints may limit the choices available to a programmer, and therefore lead to an application of merger or scenes a faire, although the court found that Unison World's choices were not practically limited in that case.[218]

F. Plains Cotton

In *Plains Cotton Cooperative Association v. Goodpasture Computer Service,*[219] the United States Court of Appeals for the Fifth Circuit reviewed the denial of a plaintiff's motion for a preliminary injunction to prevent the defendant from marketing a computer program that allegedly infringed the plaintiff's copyrighted program. In *Plains Cotton,* the plaintiff sued a group of defendants that included former employees of the plaintiff for developing a personal computer version of the plaintiff's software system for cotton farmers.[220] Because the defendants worked from a copy of the plaintiff's source code, the defendants' program was very similar to the plaintiff's on several levels.[221] The district court had denied the plaintiff's motion for a preliminary injunction, and the plaintiff appealed to the fifth circuit.[222] Reviewing the district court's findings, the *Plains Cotton* court noted that the grant or denial of a preliminary injunction was a matter for the district

---

right Office, implemented in 1988, that a single copyright registration protects all expressive elements in a computer program seems to validate *Broderbund*'s position on this point. *See* 53 Fed. Reg. 21,817 (1988).

For a discussion of other cases holding that the separable expressive elements contained in functional articles are copyrightable, *see supra* notes 119-122 and accompanying text.

216. *See Broderbund,* 648 F. Supp. at 1132.

217. *See supra* notes 110-114 and accompanying text.

218. *See Broderbund,* 648 F. Supp. at 1132.

219. 807 F.2d 1256 (5th Cir.), *cert. denied,* 484 U.S. 821 (1987).

220. *Id.* at 1258.

221. *Id.* at 1258-59. For an explanation of the difference between source code and object code, *see supra* text accompanying notes 71-73.

222. *Id.* at 1259.

court's discretion and that the district court's finding rested on its judgment about the extent of copyright protection for computer programs.[223] The *Plains Cotton* court declined to define a standard of protection on the incomplete record before it.[224] The court did, however, state that the fact that market factors played a key role in dictating the sequence and organization of any cotton market program supported the inference that those patterns might constitute "ideas."[225]

Although the *Plains Cotton* decision is necessarily a narrow one (the court was only checking for an abuse of discretion below), it does contain a significant concept. The fifth circuit's recognition that external market factors may limit the choices of structures available to a programmer and thus limit copyright protection is an implicit application of the scenes a faire doctrine. Under this doctrine, when the choices of available expression are limited by the work's theme or subject, copyright protection is also limited.[226]

G. *Softklone*

In *Digital Communications, Inc. v. Softklone Distributing Corp.*,[227] the United States District Court for the Northern District of Georgia addressed the degree of protection offered by a copyright registration for a program's display screens. The plaintiff in *Softklone* had developed a cross-talk system, which enabled the user's computer to communicate with other computers.[228] According to the court, the distinctive "status screen" of the plaintiff's program was a unique element that enabled it to achieve market popularity.[229] The plaintiff obtained separate copyright registrations for the user's manual, the program itself, and the main menu "status screen."[230] The defendant, Softklone, developed a "clone" program which, among other elements, employed the same status

223. *Id.* at 1257, 1262.

224. *Id.* at 1262.

225. *Id.*

226. *See supra* text accompanying notes 123-125 for a discussion of the scenes a faire doctrine.

227. 659 F. Supp. 449 (N.D. Ga. 1987).

228. *Id.* at 452.

229. *Id.* For a discussion of the importance of user interface elements to the marketability of a program, *see supra* notes 81-83 and accompanying text.

230. *Id.* at 453. These registrations were granted before the Copyright Office announced its policy of only accepting one registration to protect all of the elements of a computer program. *See* 53 Fed. Reg. 21,817 (1988).

screen as the plaintiff's program.[231] Initially, the *Softklone* court addressed the protection of a computer program's screen displays by the program's copyright.[232] The court noted that although *Whelan* had stopped short of extending a program's copyright to protect its screen displays, the *Broderbund* court had concluded that a program's copyright did extend to its overall structure, including screen displays.[233] The *Softklone* court rejected *Broderbund*'s holding as "an overexpansive and erroneous reading of *Whalen* [sic]."[234] Because the same screen display could be generated by two or more entirely different programs, the *Softklone* court concluded that the copyright of a program could not extend to the screen displays generated by that program.[235]

Having decided that the use of the plaintiff's status screen did not infringe the plaintiff's copyright in the Crosstalk program, the *Softklone* court next examined the plaintiff's separate copyright in the status screen itself. Rejecting the defendants' merger defense, the court found that the screen display was copyrightable, because certain aspects of it were unrelated to the way in which the computer operated and therefore constituted expression.[236] After a discussion of Softklone's defenses, the court concluded that the arrangement of Digital's main menu screen was neither an idea nor the only means of expressing an idea, and that the screen was

---

231. *Softklone*, 659 F. Supp. at 453. The defendant called its program "Mirror." *Id.*

232. *Id.* at 455.

233. *Id.* (citing Whelan Assocs. v. Jaslow Dental Laboratory, 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127 (N.D. Cal. 1986)).

234. *Id.* In the court's opinion, *Whelan* dealt only with the evidentiary use of copying of screen displays. That is, *Whelan* held that evidence of similarity of screen displays provided indirect evidence of copying the plaintiff's program. The *Whelan* court did not, according to the *Softklone* court, conclude that screen displays were protected by a program's copyright. *See id.* at 455-56 (discussing cases).

235. *Id.* at 455. The court reasoned that screen displays are not direct copies of the literary content of the program. *Id.*

236. *Id.* at 458-59. For a discussion on the copyrightability of expressive elements of a functional work, *see supra* notes 119-122 and accompanying text.

The *Softklone* court distinguished *Synercom*, stating that in that case, the input formats which were found to be uncopyrightable were relevant to the functioning of the computer. *Id.* at 460 (discussing Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003 (1978)). In the case before it, however, the court found that the arrangement of the status screen involved "considerable stylistic creativity." *Id.*

therefore copyrightable.[237]

Turning to the question of whether the defendant's status screen infringed Digital's copyrighted screen, the court concluded that there was substantial similarity.[238] Although the *Softklone* court purported to employ the bifurcated test for substantial similarity, it made no analysis of either determination, contenting itself with conclusory statements that substantial similarity between both ideas (the extrinsic test) and expression (the intrinsic test) of the two screens existed.[239]

Clearly, under the new Copyright Office policy allowing only one registration for all of the elements of a computer program, the basis for *Softklone*'s holding that a program's copyright does not protect its display screens is no longer valid.[240] *Softklone*'s recognition, however, that the program's display screens involved protectable creative effort is important. After the Copyright Office's policy announcement in 1988, the protection that the *Softklone* court found available for the display screens as "audiovisual works" should now be available to the same extent under the copyright registration for the program.[241]

H. Cams

In 1989, after the change in Copyright Office policy, the United States District Court for the District of Connecticut addressed the copyrightability of computer screen displays in *Manufacturers Technologies, Inc. v. Cams, Inc.*[242] While working for the plaintiff, two of the defendants in *Cams* had developed a program called "Costimator," which enabled the user to estimate the cost of machining a manufactured part.[243] The plaintiff obtained copyright registrations for the program, the manual, and the screen displays.[244] Subsequently, the defendants developed a program to

237. *Id.* at 463.
238. *Id.* at 464-65.
239. *Id.* at 465.
240. *See* 53 Fed. Reg. 21,817 (1988).
241. The Copyright Office stated that one registration under the class that predominated a computer program would protect all copyrightable elements in that program, regardless of their class. *See* 53 Fed. Reg. 21,817. For a discussion of the different classes of copyrightable works, *see supra* note 92 and accompanying text.
242. 706 F. Supp. 984 (D. Conn. 1989).
243. *Id.* at 987.
244. *Id.* at 988.

compete with the plaintiff's.[245] Analyzing the plaintiff's infringement claim, the *Cams* court noted at the outset that the extent to which a copyright for a program extended to its screen displays was unclear.[246] Adding to the confusion was the fact that the Copyright Office would no longer accept separate registrations for computer screen displays, stating instead that the copyright for a program extended to both literal and non-literal elements of the program, including its screen displays.[247] The *Cams* court found little help from precedent. The only two cases to directly discuss copyright protection of screen displays were *Broderbund* and *Softklone,* and the *Cams* court found itself unable to rely on either of these. First, the court stated that the *Softklone* opinion was of questionable validity, because it based a finding of infringement on a separate registration for screen displays.[248] The change in practice by the Copyright Office now required all elements of a program to be protected by one registration, and thus *Softklone*'s holding that a program's copyright did not extend to its screen displays would prevent the protection of any expression contained in those displays.[249] The *Cams* court was not persuaded by the *Broderbund* approach, either, however, stating that *Broderbund* appeared to be based on a misapplication of *Whelan.*[250]

Left without the aid of a precedent, the *Cams* court decided that a copyright registration for a program should protect both the literal elements of the program itself and the non-literal elements of its screen displays, user interface, and structure.[251] The court found it helpful to create the legal fiction of two separate registrations: one for the program, and one for the screen displays.[252] In this way, the *Cams* court found itself able to focus on the copyrightable expression of each idea and to avoid confusing the sepa-

245. *Id.* at 989. There was evidence that in addition to the defendants' general knowledge of the structure of the plaintiff's program, the defendants had photographs of the screen displays of plaintiff's program to work from. *Id.* at 988.

246. *Id.* at 990.

247. *Id.* at 990-91.

248. *Id.* at 992 (discussing Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449 (N.D. Ga. 1987)).

249. *Id.* (discussing *Softklone*).

250. *Id.* (discussing Broderbund Software, Inc. v. Unison World, 648 F. Supp. 1127 (N.D. Cal. 1986)). For a discussion of criticisms of the *Broderbund* opinion, *see supra* note 214 and accompanying text.

251. *Id.* at 993.

252. *Id.*

rate ideas.[253] The court went on to determine that both the "flow" of the plaintiff's screen displays and certain elements of their layout were protected by the fictitious copyright registration for the screens.[254]

Having set out the copyrightable elements of the plaintiff's screen displays, the court turned to the determination of whether the defendant had infringed the plaintiff's. The court purported to employ the traditional bifurcated substantial similarity test.[255] First, the court compared only the non-protected aspects of the works to determine whether there was copying.[256] Having determined that there was sufficient similarity between the unprotected elements of the works to lead to an inference of copying, the court then performed the intrinsic inquiry, evaluating the protected elements of the works from the standpoint of an ordinary observer to determine whether there was substantial similarity.[257] The court ultimately concluded that there was infringement and granted a permanent injunction.[258]

The *Cams* court made a practical attempt to reconcile the *Softklone* decision with current Copyright Office practice. Unfortunately, the *Cams* court's attempt to analyze the literal and non-literal elements of the program separately served only to demonstrate the wisdom of the Copyright Office's policy decision that one registration should protect all elements of a program. The *Cams* court hoped that creating the legal fiction of a separate registration for the screen displays would help to clarify analysis.[259] The court's own analysis, however, demonstrates that this legal fiction simply is not helpful.[260] Because the literal and non-literal elements of a

---

253. *Id.*

254. *Id.* at 994-98. The court's analysis here refutes its claim that creating a legal fiction of a separate registration will avoid confusion of ideas. The court's protection of the "flow" of screens could more properly be characterized as protection of the sequence of the underlying program, and the protection of the layout of the screens as actual protection of the screen displays. *See id.*

255. *Id.* at 1000.

256. *Id.* Under a traditional extrinsic analysis, it is important to compare the two works as a whole, not just the unprotected elements of the two works, in making this inquiry. *See supra* note 130 and accompanying text.

257. *Id.* at 1001. At this phase, the court properly excluded unprotected elements from consideration. *Id.*

258. *Id.* at 1002.

259. *Id.* at 993.

260. For a discussion of how the court confused protectable audiovisual elements of the screen displays with the structure of the program, *see supra* note

program are necessarily interrelated, it is not possible to separate them entirely when conducting an infringement analysis.[261]

Additionally, it appears that the *Cams* court misapplied the extrinsic inquiry called for by the substantial similarity test. For this inquiry, the court compared only unprotected elements of the works.[262] Because the *Cams* court recognized that the purpose of the extrinsic inquiry was merely to determine whether copying may have occurred, and not to determine infringement, this misapplication was probably harmless error.[263] At least one commentator, however, feels that the *Cams* court mistakenly based a finding of infringement on similarities between some of the unprotected elements of the two programs because of its misapplication of the extrinsic test.[264]

I. Lotus

In one of the most recent opinions to address the scope of copyright protection for non-literal elements of a program, the United States District Court for the District of Massachusetts concluded that such elements could be protected in *Lotus Development Corp. v. Paperback Software International.*[265] The plaintiff in *Lotus* claimed that the defendant's program, VP-Planner, infringed certain non-literal elements of its spreadsheet program, Lotus 1-2-3.[266] After a thorough analysis of both the constitutional and the statutory bases for protecting programs, the *Lotus* court concluded that neither directly addressed the protection of non-literal program elements.[267] The *Lotus* court therefore turned additionally to the legislative history of the Copyright Act and to the CONTU report for guidance, concluding from these that Congress intended to make traditional copyright doctrines, such as the idea/

254.

261. *See* Comment, *supra* note 93, at 537-38 (program and screen displays related and dependent upon one another).

262. The extrinsic inquiry calls for the court to examine the two works as a whole in order to determine whether copying occurred. *See supra* note 130 and accompanying text.

263. The court explicitly stated that a finding of infringement could be based only on a finding of substantial similarity of *protected* elements under the intrinsic inquiry. *Cams*, 706 F. Supp. at 1000.

264. *See* Comment, *supra* note 93, at 560-62.

265. 740 F. Supp. 37 (D. Mass. 1990).

266. *Id.* at 42.

267. *Id.* at 47.

expression dichotomy, applicable to computer programs.[268] Having decided to apply this analysis, the *Lotus* court next had to determine which parts of the plaintiff's program were protectable expression.[269]

The court began its determination by rejecting the defendant's arguments that Lotus' user interface was not copyrightable because it was primarily functional, like the "figure-H" pattern discussed by the *Synercom* court.[270] The *Lotus* court credited expert testimony that the creation of a user interface required considerable creativity and originality.[271] Further, the *Lotus* court stated that simply by being "useful," a program did not lose copyright protection.[272] Rather, elements of expression in a useful article are still copyrightable, if they can be recognized independently of the function of the article.[273]

Having rejected Paperback's "functionality" argument, the *Lotus* court went on to set forth a three-part test for determining the copyrightability of the elements of a computer program. First, according to the court, the decisionmaker must define the "idea" in order to separate it from protectable expression.[274] In order to accomplish this task, the *Lotus* court recommended using Judge Hand's abstractions test.[275] The second step in determining

---

268. *Id.* at 54. The *Lotus* court also noted that leaving trade secret law as the only form of legal protection for a user interface would be an unsatisfactory solution. *Id.* at 56. For a discussion of the disadvantages of trade secret protection for programs, *see supra* text accompanying notes 42-53.

269. *Id.* at 54-55.

270. *Id.* at 55. For a discussion of the reasoning of the *Synercom* opinion and an explanation of Judge Higginbotham's "figure-H" analogy, *see supra* text accompanying notes 133-144.

271. *Id.* at 56. For additional discussion of the originality and expense that go into the creation of a user interface, *see supra* text accompanying notes 80-83.

272. *Id.* at 57. The court stated that:

> It does not follow that when an intellectual work achieves the feat of being useful as well as expressive and original, the moment of creative triumph is also a moment of devastating financial loss - because the triumph destroys copyrightability of all expressive elements that would have been protected if only they had not contributed so much to the public interest by helping to make some article useful.

Id.

273. *Id.* at 58. For a discussion of other cases allowing copyright protection for the expressive elements contained in useful articles, *see supra* notes 119-122 and accompanying text.

274. *Id.* at 60.

275. *Id.* For an explanation of the abstractions test, *see supra* text accompa-

copyrightability requires the decisionmaker to determine whether the expression contains only elements essential to the expression of the idea or instead contains elements not essential to that idea's expression.[276] Third, the *Lotus* court stated that the decisionmaker should determine whether those elements of the expression that were non-essential made up a substantial part of the work.[277]

Applying this test, the *Lotus* court characterized the underlying idea of the program as an electronic spreadsheet.[278] Although the idea of a spreadsheet was not copyrightable, the court stated that varying expressions of that idea could be protected, noting that there were already several other programs on the market that expressed the spreadsheet idea in different ways.[279] Next, the court determined that certain elements of the plaintiff's program, including the command structure, were not essential to the expression of the idea of a spreadsheet.[280] Because parts of Lotus' program were not essential to the expression of the spreadsheet idea, and because those parts were a substantial part of the program, those parts were protectable.[281] Further, the *Lotus* court stated that because Paperback's program was substantially similar to the plaintiff's, its program infringed the plaintiff's copyright.[282]

The *Lotus* court attempted to fashion a much-needed principled test for determining the copyrightability of various program elements. Although the general ideas behind the court's test are

---

nying notes 107-109.

276. *Id.* at 61. This step is similar to the merger doctrine. It is more narrow than merger, however, because merger also takes into consideration instances where the expression may not be essential to the idea, but nevertheless the ways of expressing the idea are so limited that merger should apply. For a discussion of the merger doctrine, *see supra* text accompanying notes 110-114.

277. *Id.* at 61. The requirement that a "substantial" part of the work be expressive in order to deserve copyright protection seems to be a more difficult standard than the Copyright Act's minimal "originality" requirement. For a discussion of that requirement, *see supra* text accompanying notes 98-100.

278. *Id.* at 65.

279. *Id.*

280. *Id.* at 68. This is the second step of the court's three-part test.

281. *Id.* at 68.

282. *Id.* at 70. The court did not endorse a specific test for infringement, stating instead that from the perspective of both an expert and an ordinary observer, the works were substantially similar. *Id.* The court may have found a circumstantial test unnecessary, because the defendant had stipulated to copying the plaintiff's program, and the court therefore had sufficient evidence of direct copying to find infringement without the substantial similarity test. *See id.* at 68-69.

logical, its test as applied may be too narrow. The court's recognition that Judge Hand's abstractions test may be useful in separating idea from expression in programs is laudable.[283] The court's application of the abstractions test, however, fails because it attempts to define only one idea for the work.[284] The purpose of the abstractions test is to gradually take the work to more general levels, at which parts of the work will become unprotectable ideas; thus, the test implicitly recognizes that every work contains a number of ideas.[285] By attempting to isolate a single idea, the *Lotus* creates the same risk that the *Whelan* court did of offering overbroad protection.[286] The second step of the *Lotus* test is theoretically similar to merger. Because that test is narrower than the traditional merger doctrine, it may be necessary to recast that step of the test.[287] Additionally, the third step of the *Lotus* test may place a greater demand on works than the Copyright Act itself. Because the Act only requires minimal original contributions from the author in order for a work to merit protection, the *Lotus* test may be too demanding.[288]

V. NON-LITERAL PROGRAM ELEMENTS MERIT PROTECTION

A. *The Need for Protection to Encourage Innovation*

Clearly, some level of protection is required for the non-literal elements of a program. The non-literal elements are a significant, perhaps the most significant, part of a program. The user interface has been identified as the most important feature in determining the marketability of mass-marketed computer programs.[289] Because the average user only communicates with the computer through the program's menus and screen displays, it is easy to see why the user interface is so crucial to the success of a program.[290]

---

283. For a discussion of how this test adapts well to programs because of the manner in which they are written, *see supra* note 108 and accompanying text.

284. *See id.* at 65.

285. *See supra* note 109 and accompanying text.

286. For a discussion of the problems created by *Whelan*, *see supra* text accompanying notes 195-201.

287. *See supra* note 276 for an explanation of the differences between the two standards.

288. For a discussion of the minimal nature of the originality requirement under the Copyright Act, *see supra* text accompanying notes 98-100.

289. *Clones*, *supra* note 10, at 195.

290. *See Screen Displays*, *supra* note 15, at 1132 (typical user's only contact with computer is through screens).

A look at history verifies this fact. In the early years of computers, when the only people using them were computer scientists who were highly trained in mathematics and engineering, the only "communication" with computers took place through long rolls of punched tape.[291] It was not until the early 1980's that simpler user interfaces began to appear on programs marketed to the public.[292] During the last decade, one writer has noted, the user interface has become a key consideration in designing programs.[293] The design of user interfaces has become so important that the new science of Computer-Human Interaction has sprung up around it.[294]

Because the user interface of a program is so important to that program's success on the market, programmers invest a great deal of time and creativity in the design of better, easy-to-use interfaces. The user interface may even be written separately from the program.[295] Often, the interface is written by a team made up of psychologists, engineers, and other specialists.[296] Currently, one writer estimates, at least 40 percent of the instructions in a program are written for the user interface.[297] There is no question that designing interfaces of this caliber requires the investment of a substantial amount of effort.[298]

Although there are some definite goals to achieve, the effort involved in creating a user interface is not limited to the application of mechanical rules and formulas; it involves a great deal of creativity.[299] In fact, the choices made by a programmer are gener-

---

291. Curtis, *supra* note 10, at 51.

292. *Id.* at 52.

293. *Id.* For example, the "mouse," a manual device for positioning the cursor on the screen, was patented in 1967, but it was not widely marketed until 1983. *Id.*

294. For a discussion of this new science, *see supra* text accompanying note 82.

295. Curtis, *supra* note 10, at 62 (user interface design has emerged as separate discipline).

296. *Screen Displays, supra* note 15, at 1132.

297. Curtis, *supra* note 10, at 53, 56.

298. *Clones, supra* note 10, at 216. For example, the design of the interface for the Xerox Star system required about thirty work-years of programmer time. Much of this time was invested before the computer hardware was even built or a line of software was written. *Id.* at 216 n.158. *See also Screen Displays, supra* note 15, at 1152.

299. Some of the goals of user interface design include (1) minimizing user learning time, (2) maximizing speed of performance, (3) minimizing the rate of user errors, (4) maximizing user satisfaction, and (5) maximizing users' retention of knowledge over time. Menell, *supra* note 1, at 1054.

ally said to be influenced by his training, experience, and individual writing style.[300] The way in which a programmer chooses to break down a task, or the logical "flow" of a program, can reflect a creative choice.[301] At least one commentator has recognized that the sequence and combination of the subtasks of a program can represent creative expression.[302] Programmers themselves state that programming is as much an art as it is a science.[303]

Although user interfaces are expensive, in terms of time and money, to produce, they are quite inexpensive to copy. Imitation has been called "technically effortless" in the software field.[304] CONTU noted in their final report that the cost of duplicating software was far less than the cost of creating it.[305] In fact, it is possible to copy a computer program in only a few minutes and to reproduce as many copies as one wishes.[306]

Because the costs of producing a user interface are extremely high when compared to the costs of copying one, it is essential to offer some protection to the creative elements of user interfaces in order to encourage more developments and innovation in this field.[307] Without protection, the author of a user interface would not have sufficient "lead time" on the market to recoup his development costs before imitators began profiting from his creative efforts by marketing programs with similar interfaces.[308] Some pro-

---

300. *Binary Bards, supra* note 7, at 1529. Each programmer develops her own style of expression. *Id.* at 1535.

301. For a discussion of the different ways to break down the task to be performed by a program, *see supra* text accompanying notes 67-70.

302. *Binary Bards, supra* note 7, at 1533.

303. *Id.* at 1538. One programmer described the programming process as follows:

> Writing microcode is like nothing else in my life. For days there's nothing coming out. The empty yellow pad sits there in front of me, reminding me of my inadequacy. Finally, it starts to come. I feel good. That feeds it, and finally I get into a mental state where I'm a microcode-writing machine. . . . After a while, you're like a kid on a jungle gym. There are all these constructs in your mind and you can swing from one to the other with ease.

Id. at 1537 (quoting T. KIDDER, THE SOUL OF A NEW MACHINE 101-02 (1981)).

304. *Id.* at 1509.

305. CONTU Report, *supra* note 14, at 22.

306. *Binary Bards, supra* note 7, at 1509.

307. Recall that promoting the progress of science is the goal of copyright law. *See supra* text accompanying notes 84-87.

308. *Binary Bards, supra* note 7, at 1509. For a discussion of the importance of allowing a programmer to recoup development costs, *see supra* note 86.

tection must be offered to encourage developers to expend the effort required to design new interfaces.[309] The developer must be guaranteed sufficient protection to allow him to recover the costs of developing a unique interface.[310]

B. The Standardization Argument

Opponents of protection for non-literal elements of computer programs argue that copying is necessary to foster standardization in the programming industry. Because the costs of learning to operate a new program are high, these commentators argue that the public would be better served by compatible user interfaces which would allow operators to learn new programs quickly.[311] This argument overlooks the way in which most interfaces become "standards" and fails to take into account some of the costs of standardization. Although it is true that having an industry standard decreases the amount of used training time, this savings may come at the cost of locking the industry into an inefficient standard.[312] Once a standard has been achieved, it will likely slow innovation, because it is difficult to depart from that standard.[313] A good example of this problem is the QWERTY keyboard commonly used by typists today. The configuration of this keyboard was historically designed to slow down one-finger typists to prevent them

---

309. *Clones, supra* note 10, at 216.

310. *Binary Bards, supra* note 7, at 1500.

311. *See, e.g.,* Curtis, *supra* note 10, at 53 (cost of learning software package substantial); Damman, *supra* note 85, at 432 n.75 (costs $1,000 to train worker on Lotus 1-2-3).

312. Farrell, *supra* note 16, at 37.

313. *Id.* Farrell illustrates the problem of departing from an accepted standard in this way:

> [T]hink of a generic Western movie. At sundown, the cowboys are in the middle of the desert, with no trees to tie the horses to. So does one of them stay up all night and hold the horses? No, they just tie the horses to one another, and go to sleep! The horses could go anywhere, but in the morning they have wandered only a few hundred feet. Why not five miles? Well, imagine yourself as an adventurous horse, tied to several others. When you want to explore, the others are eating or taking a nap, so you yank on the ropes and they say, in horse language, 'Oh, you wanted to go somewhere? Wait a bit, and maybe I'll be ready to go soon.' But by that time you are interested in this piece of cactus over there, and so as a group you never get far. . . . Similarly, it can be hard for users or vendors to coordinate a switch from an old standard to a new one, even if all would like to do so.

Id.

from jamming the hammers of early typewriters by striking two keys in quick succession.[314] This configuration does not promote speed on modern typewriters.[315] In fact, studies indicate that the Dvorak Keyboard System would be much more efficient in terms of speed and accuracy, but because of training costs, virtually all companies remain with the antiquated standard.[316] Fostering standardization in the computer industry may likewise tie it to inefficient interfaces and retard innovation.

Because the standard chosen by an industry is not necessarily the one that is the most efficient, one must question how a standard can be formed. One commentator has noted that while research and development costs total only about 15 percent of the total revenues of the software industry, marketing costs account for 35 percent.[317] It seems logical that the only way for a software firm to encourage its product to become a "standard" in the buying community, even if that product is not the most efficient, is for that firm to expend substantial amounts on marketing that program and on training the public to use it.[318] Once a company has set a standard through its marketing efforts, it should not be forced to share the benefit it acquired through these expenditures with other companies. Put another way, every programmer is free to work to make his program the industry standard. The fact that the public has been "trained" to another standard should not justify free copying of that standard.[319]

Because the creation of a program's non-literal elements represents a substantial expenditure, some form of protection is required to encourage innovation by allowing creators to recoup their development costs. Further, there is no justification for requiring these creators to share their developments with competitors cost-free in the name of standardization. There are, however, a number

314. Menell, *supra* note 1, at 1067.

315. *Id.*

316. *Id.*

317. Menell, *supra* note 1, at 1065.

318. For example, the plaintiff in *Synercom* only achieved market success after substantial investment in training its customers to use its program. The court estimated the cost of this training at $500,000. Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003, 1008 (N.D. Tex. 1978).

319. If the programmer does not wish to expend the effort to create a new standard, he is free to negotiate a license with the company that has achieved the current market standard, or to write an application program compatible with that company's existing program.

of limits on the choices available to programmers, and providing too much protection to creators may give them monopoly-like power in certain areas.[320] In order to avoid giving too much protection to creators, thereby stifling competition, it is necessary to develop a principled means of separating the situations in which there are a wide range of choices available from those in which the choices are limited by practical or mechanical considerations. In determining the scope of copyright protection, courts should grant copyright only to those elements of a program that are truly creative expression. The "successive filtering test" developed by David Nimmer represents a recent attempt to accomplish this task.

VI. THE SUCCESSIVE FILTERING TEST - A STEP IN THE RIGHT DIRECTION

Courts seem to be in general agreement that to the extent they contain creative expression, computer programs ought to be protected. The problem for these courts, however, is separating expression from idea and identifying instances in which the alternative means of expression are narrowly limited. Recognizing that computer programs are difficult for courts and juries to understand, the "successive filtering" approach, developed by David Nimmer, provides an infringement analysis intended to assist courts in determining which elements of a program are entitled to copyright protection.[321] Under this approach, courts are to use expert testimony to "filter out" unprotectable elements of a program and then to compare the remaining protectable elements in determining whether the program's copyright has been infringed. To accomplish this, a different copyright doctrine is applied at each level of the test to remove unprotectable material, ending up with a "core" of protectable expression.[322] This test seeks to minimize confusion by removing elements that are unprotectable from consideration and by conducting the analysis under well-established copyright principles.[323]

---

320. For a discussion of some of these limits, *see supra* text accompanying notes 78-83.

321. *Structured Approach, supra* note 15, at 625; 3 NIMMER, *supra* note 34, § 13.03[F].

322. 3 NIMMER, *supra* note 34, § 13.03[F], at 13-62.24.

323. *Structured Approach, supra* note 15, at 635.

A. *Application of Successive Filtering*

1. *Idea v. Expression*

Under the first step of the successive filtering test, courts are called on to separate the ideas of a program, which are unprotectable, from the program's expression.[324] Although this determination is at best a difficult one, courts and commentators have noted that Judge Hand's abstractions test can be useful.[325] The "top-down" approach to computer programming makes Judge Hand's test particularly applicable, because that test, which moves from specific expression towards more general levels, simply disassembles a program in the reverse order of the way it was plotted.[326]

At this stage of the analysis, it is important for courts to be aware that a program may contain any number of "ideas."[327] Judge Hand's abstractions test recognizes that a work may contain numerous ideas.[328] In the past, however, a number of courts have tried to identify a single idea in a computer program in order to separate expression from it.[329] This reasoning leads to overbroad

---

324. 3 NIMMER, *supra* note 34, § 13.03[F][1]; *Structured Approach, supra* note 15, at 636-40.

325. 3 NIMMER, *supra* note 34, § 13.03[F][1], at 13-62.27; *Structured Approach, supra* note 15, at 636-37; *Binary Bards, supra* note 7, at 1568; *Process or Expression?, supra* note 18, at 899; Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 60 (D. Mass. 1990); Computer Assocs. Int'l Inc. v. Altai Inc., 20 U.S.P.Q.2d 1641, 1651 (E.D.N.Y. 1991).

326. *See* 3 NIMMER, *supra* note 34, § 13.03[F][1], at 13-62.27; *Structured Approach, supra* note 15, at 638. For a discussion of the "top-down" approach to program writing, see *supra* text accompanying notes 63-70. For a discussion of Judge Hand's abstractions test, *see supra* text accompanying notes 107-109.

327. *See Process or Expression?, supra* note 18, at 901. The writer notes that "[w]hile the possibility that only certain aspects of a play may constitute protected expression is not at all controversial, courts have unanimously treated the protection of a program's structure as an all-or-nothing proposition." *Id. See also Clones, supra* note 10, at 205; Comment, *supra* note 93, at 548-49. This writer warns: "Courts are cautioned that this step must include a thorough analysis of all ideas which are included in a program-not simply the major idea." *Id.*

328. *See* 3 NIMMER, *supra* note 34, § 13.03[F][1], at 13-62.28 (abstractions test implicitly recognizes that work may contain mixture of numerous ideas); *Structured Approach, supra* note 15, at 640 (same); *see also supra* note 109.

329. *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1236 (3d Cir. 1986) (idea of program is its purpose), *cert. denied*, 479 U.S. 1031 (1987); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 65 (D. Mass. 1990) (idea of program is spreadsheet); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (idea of program is printing cards).

protection of all elements not "essential" to the main purpose of a program.[330] Careful adherence to the abstractions test will prevent this arbitrary distinction and foster more predictable results.[331]

2. *Merger*

Once the elements of a program that represent unprotectable ideas have been excluded from consideration, courts, through expert testimony, are next asked to identify the instances in which the available choices of expression are so limited that the merger doctrine should be applied.[332] Where the idea expressed is one that can only be stated in a narrowly limited number of ways due to logic or efficiency considerations, merger should be applied to limit copyright protection accordingly.[333] For example, the idea of placing the file name and cursor status line at the bottom of a display screen in a word processing program would be denied protection under the merger doctrine, because the nature of a word processing program requires the cursor to remain at the bottom of the screen most of the time in order to allow user to view on the screen what he has just written.[334] Because efficiency practically requires the cursor status line to be displayed at the bottom of the screen in that instance, merger would apply. When applying merger, it is important for courts, with expert assistance, to recognize when choices are limited for all practical purposes. Although an infinite number of ways may theoretically be available to express an idea, considerations of efficiency may make all but a few of these choices impractical; and merger should be applied in such instances.[335] Expert testimony should help to clarify when efficiency considerations really do impose practical limitations on expression. The

---

330. *See Whelan*, 797 F.2d at 1236.

331. Although, inevitably, as in all other copyright cases, drawing the idea/expression line will be an "ad hoc" determination. *See supra* text accompanying note 105.

332. 3 NIMMER, *supra* note 34, § 13.03[F][2]; *Structured Approach, supra* note 15, at 640-42. For a discussion of the merger doctrine, see *supra* text accompanying notes 110-114.

333. 3 NIMMER, *supra* note 34, § 13.03[F][2], at 13-63, 13-65; *Structured Approach, supra* note 15, at 640-42. For a discussion of some of these limits, *see supra* text accompanying notes 78-83.

334. Damman, *supra* note 85, at 430.

335. *See* 3 NIMMER *supra* note 34, § 13.03[F][2], at 13-64. As Nimmer states, the fact that two programs use the most efficient method of achieving a result should not be evidence of copying. *Id.*

merger doctrine should not be applied so broadly that only inefficient means of achieving a result are protected.[336]

3. Scenes a Faire

In the next step of the successive filtering test, courts should exclude program elements that are dictated by external considerations.[337] This step of the analysis can be equated to the scenes a faire doctrine, which denies copyright protection to elements that are an inherent part of a work's theme, or that are standard devices used in expressing that theme.[338] In this step, courts, again with expert assistance, should exclude elements of a program that are dictated by standard computing techniques.[339] Elements excluded by this step include any choices dictated by hardware standards, manufacturers' standards, or standards which have in fact become widely shared among programmers.[340]

4. Public Domain

Because material in the public domain is not protected even when integrated into a copyrighted work, it is excluded from consideration at this stage of the test.[341] This phase of the test is espe-

---

336. *Cf.* Menell, *supra* note 1, at 1086-87. Menell contends that only inefficient programming methods should be protectable, but he acknowledges that this brings about the perverse result of requiring a plaintiff to argue that his program is inefficient in order to receive copyright protection. *Id.*

The mere fact that a program is efficient should not disqualify it from protection. Only where efficiency demands one of a narrow range of choices should merger apply. If other means are less efficient, but still practical, merger should not apply. Expert testimony will be especially important in identifying those situations in which efficiency and other concerns really do impose narrow limits on the choices of expression.

337. 3 NIMMER, *supra* note 34, § 13.03[F][3]; *Structured Approach, supra* note 15, at 642-48.

338. For a discussion of the scenes a faire doctrine, *see supra* notes 123-125 and accompanying text.

339. 3 NIMMER, *supra* note 35, § 13.03[F][3], at 13-65; *Structured Approach, supra* note 15, at 642-43. Nimmer notes that in certain situations, it is nearly impossible to write a program without using techniques which are standard for implementing a certain function. *Id.* The *Plains Cotton* court also recognized that programming choices dictated by external market considerations should not be protected. Plains Cotton Coop. Assoc. v. Goodpasture Computer Serv., 807 F.2d 1256, 1262 (5th Cir.), *cert. denied*, 484 U.S. 821 (1987).

340. For a more complete listing of excludable elements, *see* 3 NIMMER, *supra* note 34, § 13.03[F][3].

341. 3 NIMMER, *supra* note 34, § 13.03[F][4]; *Structured Approach, supra*

cially important, because of the "stepping stone" method of development in the software industry.[342] Because programmers often share ideas and build on existing routines, it should not be unusual to find some similarities between programs incorporating "borrowed" parts.[343] Careful exclusion of those borrowed elements which are not protected by copyright will avoid the possibility that a court will find infringement based on similarities between the unprotected portions of two programs.

B. Problems with Successive Filtering

Although the successive filtering approach does have the advantage of allowing courts to employ familiar copyright doctrines when conducting an infringement analysis, the test by itself creates some potential difficulties and sacrifices unnecessarily some of the advantages of the traditional bifurcated test for substantial similarity. The interaction between courts and expert witnesses called for by the successive filtering approach creates the danger of too much reliance on expert testimony. Because successive filtering calls for a single inquiry into substantial similarity, it will be difficult for courts to exclude expert opinions on substantial similarity between the "core" protected elements of programs after relying on expert testimony to sift out the unprotected elements. The workings of computer programs can be complex and difficult for a court to understand. The court in *Q-Co Industries v. Hoffman*,[344] discussed in Part IV above, for example, openly acknowledged the difficulties it had in comprehending the technical workings of computer programs.[345] It is therefore quite conceivable that a court might be tempted to abdicate its decisionmaking responsibility entirely and rely instead on expert testimony, especially after extensively employing such testimony for successive filtering. The opinion in *E.F. Johnson v. Uniden Corp. of America*,[346] analyzed and criticized above in Part IV, provides an example of this sort of danger. Although it did not totally abdicate its decisionmaking re-

---

note 15, at 648-49.

342. For a discussion of the practice of building new programs on pre-existing ones, *see supra* text accompanying notes 74-77.

343. 3 NIMMER, *supra* note 34, § 13.03[F][4].

344. 625 F. Supp. 608 (S.D.N.Y. 1985).

345. The court stated that "[t]he challenge to counsel to make comprehensible for the court the esoterica of bytes and modules is daunting." *Q-Co*, 625 F. Supp. at 610.

346. 623 F. Supp. 1485 (D. Minn. 1985).

sponsibility, the *Johnson* court did adopt a test that significantly limited the scope of protection available to non-literal program elements because it found that the ordinary observer test was difficult to apply.[347] Once a court justifies effectively limiting the scope of legal protection on the grounds that determining the proper scope is a difficult process, it is a small step for that court to turn the decision entirely over to the experts, because that process is even easier.

Even courts that are conscientious and attempt to make an independent comparison of the protected elements of programs for substantial similarity face the danger of being confused or influenced by similarities encountered during successive filtering or of being sub-consciously influenced by the expert opinions those courts relied on in performing successive filtering. Courts are obviously influenced by expert testimony, and it may be difficult for a court that has heard such testimony to credit it for one part of successive filtering, but disregard it for another part of the same inquiry.[348] The opinion in *Manufacturers Technologies, Inc. v. Cams, Inc.*,[349] discussed in Part IV above, demonstrates how a court can become confused and possibly influenced by expert testimony on similarity of unprotected program elements. Although it purported to base its finding of infringement solely on the similarities between the protectable elements of the programs at issue, the *Cams* court appears not to have managed to do so.[350] In spite of its intentions to the contrary, the *Cams* court based its finding of infringement partially on similarities between elements that the court found to be unprotected.[351] The *Cams* court had a clear understanding of the applicable law; it was simply confused by the complexity of the factual situation, and its inclusion of unprotect-

347. *Johnson*, 623 F. Supp. at 1493. For a discussion of the test adopted by the *Johnson* court and its effects on copyright protection, *see supra* text accompanying notes 153-164.

348. Courts naturally place reliance on expert testimony in such a complex area. *See, e.g., Whelan*, 797 F.2d at 1246 (court relied on plaintiff's expert for test); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 56 (D. Mass. 1990) (relying on expert testimony on creation of user interfaces); Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 994 (D. Conn. 1989) (crediting plaintiff's four experts over defendant's expert).

349. 706 F. Supp. 984 (D. Conn. 1989).

350. The court stated that a finding of infringement could be based only on a finding of substantial similarity between the protected elements of the programs. *Cams*, 706 F. Supp. at 1000.

351. *See* Comment, *supra* note 93, at 560-62.

able elements in its finding of infringement indicates that the court was influenced sub-consciously by the expert testimony on the similarities of these unprotected elements.

Additionally, the successive filtering approach loses the proper "ordinary observer" focus of the bifurcated substantial similarity test. Because it calls for a single inquiry into substantial similarity that relies heavily on expert testimony, the successive filtering approach fails to focus the ultimate determination of substantial similarity on the ordinary observer. The ordinary observer standard was specifically designed for literary works, such as computer programs.[352] Dissecting a program with expert assistance is inconsistent with allowing the factfinder to view the non-literal elements of that program as they were intended to run. Because a program's non-literal elements are designed largely to make the program comprehensible to the ordinary observer, the substantial similarity inquiry for those elements should be conducted from the standpoint of their intended audience, the ordinary observer. The bifurcated test carefully maintains this focus; successive filtering alone does not.

VII. INTEGRATING SUCCESSIVE FILTERING INTO THE BIFURCATED TEST

In order to address these problems, a modification of the successive filtering approach is required. Successive filtering should be integrated into the traditional bifurcated substantial similarity inquiry, allowing courts to benefit from the advantages of both. Computer programs are literary works, and the bifurcated test was designed for the evaluation of literary works.[353] Unlike other literary works, however, programs can be "read" on two levels. The "look and feel" of a program, its external workings perceived by the user, is one level of operation. Beneath this "look and feel," however, lies a complex set of programming commands, incomprehensible to the ordinary user.[354] Both levels can be "read," but by very different audiences. By maintaining both inquiries of the bifurcated test, but allowing the group qualified to make each inquiry to do so, courts can keep the advantages of the bifurcated

352. *See supra* text accompanying notes 128-132.

353. *See supra* text accompanying notes 128-132 for a discussion of the development of the bifurcated test.

354. For a discussion of source and object codes, *see supra* text accompanying notes 71-72.

test. The extrinsic inquiry should be made by courts using the assistance of experts possessing the requisite knowledge to analyze the underlying ideas and workings of programs. The intrinsic inquiry, as in any other literary infringement case, should be performed by the jury, from the standpoint of an ordinary observer. Successive filtering should be employed by the court between these two steps to assure that only protectable elements are examined by the jury. Using successive filtering ensures that this can be done without confusing or misleading the jury with expert testimony. Integrating the successive filtering test into the traditional bifurcated substantial similarity inquiry should thus lead to more principled and accurate decisionmaking in the protection of computer programs.[355]

A. *Application of the Integrated Substantial Similarity Test*

Under this integrated approach, a three-step process emerges. First, courts should perform the extrinsic branch of the traditional test, comparing the two works as a whole in order to determine whether the ideas of the works are similar in such a way that it appears the protected work was used in preparing the second work.[356] Expert testimony should be employed for this inquiry, because the ordinary trier of fact would not be qualified to tell from looking at two sets of code whether they were based on the same ideas.[357] Thus, courts should turn to qualified experts for the determination under the extrinsic branch of the test. Relying on experts at this stage of the test does not remove the ultimate decision from the jury, because the purpose of the extrinsic branch of the test is merely to make the preliminary determination of whether there is sufficient similarity between the underlying themes and ideas of the two works to conclude that the first work was consulted in preparing the second work.[358]

Next, if a court concludes, based on expert testimony, that the ideas behind that two works in issue are similar, the court should apply the successive filtering process to sift out unprotectable ele-

355. For a discussion of the bifurcated substantial similarity test, *see supra* text accompanying notes 128-132.

356. Both the protected and unprotected elements of the works are considered at this phase of the inquiry. *See supra* note 130.

357. *See* Comment, *supra* note 93, at 548.

358. For a discussion of this point, see *supra* note 130.

ments.[359] The application of various copyright law doctrines by successive filtering will remove elements which are not protected and leave a core of protectable material.[360] The court should again turn to qualified experts for this process to ensure accurate decisions. Exclusion of the jury from this phase ensures that the jury will not be influenced by permissible similarities between the unprotected elements of the works when making the ultimate determination on infringement.[361]

After experts have assisted the court in applying successive filtering, the protectable "core" should be submitted to the factfinder for a determination under the intrinsic analysis of whether, from the standpoint of an ordinary observer, the two works appear to be substantially similar.[362] This intrinsic analysis could be conducted by an examination of the screen displays, menus, flowcharts, or any other manifestation of the program's non-literal elements that would be comprehensible to the average user. The factfinder is qualified to make this determination, because any similarity in non-literal elements should be perceptible to the ordinary observer.[363] In fact, the purpose of the ordinary observer standard is to assure that a finding of infringement is based on a comparison of the works as they would be viewed by their intended audience. Thus, the jury is actually in a better position than experts are to make this determination.

B. Advantages of the Integrated Test

Integrating successive filtering into the bifurcated substantial similarity test has several important advantages. The principal advantage of this approach is that it resolves the problems encountered by courts addressing the copyrightability of non-literal elements of computer programs. Each of the difficulties encountered by the opinions discussed above in Part IV would be avoided through application of the integrated approach.[364] Additionally,

359. *See supra* notes 321-343 and accompanying text for a discussion of this process.

360. 3 NIMMER, *supra* note 34, § 13.03[F].

361. *See infra* text accompanying notes 366-369.

362. For a discussion of the intrinsic analysis, see *supra* text accompanying notes 131-132.

363. Indeed, the user interface is so crucial to a program precisely because it *is* comprehensible to the ordinary observer.

364. For a discussion of how the integrated approach avoids these problems, *see infra* text accompanying notes 370-401.

the integrated approach alleviates some of the difficulties encountered in applying the traditional bifurcated test to computer programs by modifying that test to suit the nature of the inquiries involved in the analysis of computer programs.

The integrated approach allows courts, experts, and juries each to make the determinations they are uniquely qualified to make. Because the internal workings of computer programs are incomprehensible to the average factfinder, the extrinsic branch of this test, which calls for examination of those workings, should be performed by the court, with the aid of expert testimony. Because of the special complexity of the extrinsic analysis in computer cases, excluding the jury from that phase in such cases minimizes the danger of confusing the jury and relieves experts of the burden of attempting to make their dissections comprehensible to the trier of fact. Dissection and analysis of the work by experts has always been proper under the extrinsic branch of the traditional bifurcated test, and the integrated approach merely continues this practice by allowing experts to make this determination independently.[365] Additionally, because experts are qualified to identify those instances in which programming choices are limited or dictated by external considerations, their assistance is invaluable in applying successive filtering. Because the court, however, is the body familiar with the workings of traditional copyright doctrines, the court directs the successive filtering, applying the relevant doctrines to the factual situation as explained by the experts. Finally, the jury or factfinder retains its traditional role, making the ultimate factual determination of infringement under the ordinary observer standard. Because the jury is able to perceive any similarities between protected non-literal elements that would be discernible to the ordinary observer, the jury is uniquely qualified to make this determination.

Even in simple cases, application of the traditional bifurcated substantial similarity test can be confusing to a jury, because it requires the jury to utilize expert testimony for the extrinsic branch and then somehow to "forget" that testimony for the intrinsic branch.[366] This difficulty is compounded in software in-

365. *See supra* text accompanying notes 129-130 for an explanation of the role of experts in the traditional bifurcated test.

366. Expert testimony is proper under the extrinsic branch of the traditional bifurcated test, but the intrinsic branch must be performed by the jury alone from the standpoint of an ordinary observer. *See supra* text accompanying notes 128-132.

fringement cases, where expert testimony must be extensively employed to make the workings of complex programs comprehensible to the jury. In such cases, the jury is likely to be unable to "forget" the testimony of experts on this seemingly arcane subject when performing the inquiry called for by the intrinsic branch of the test. The court in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*,[367] discussed and criticized in Part IV above, recognized this difficulty and cited it as a justification for abandoning the bifurcated test in software infringement cases.[368] The single inquiry adopted by the *Whelan* court and the single inquiry called for under successive filtering, however, avoid this difficulty only at the expense of creating a danger of undue reliance on expert testimony.[369] By compartmentalizing the two inquiries called for by the traditional bifurcated substantial similarity test and requiring each to be made independently by a different decisionmaker, the integrated approach avoids both of these dangers. The integrated approach relieves the jury of the burden of performing the mental gymnastics called for by the traditional bifurcated test and also avoids the problem of placing the jury in a position where it might be overwhelmed or unduly influenced by expert opinion. When the factfinder is called on to make the infringement determination under the intrinsic branch of the integrated test, it will be exposed only to the protected elements of each work. Thus, there is no danger that the factfinder will be influenced or misled by any permissible similarities between the unprotected elements of the works or by the expert analysis that eliminated those elements from further consideration.

Integrating successive filtering into the substantial similarity inquiry also solves the problems encountered by courts addressing the extent of copyright protection for non-literal program elements. Most importantly, perhaps, the integrated approach prevents courts from creating arbitrary definitions of the unprotectable ideas in each case. Because the way in which ideas are defined directly determines the scope of copyright protection for a work, it is important to have a principled means of identifying those

367. 797 F.2d 1222 (3d Cir. 1986); *cert. denied*, 479 U.S. 1031 (1987).

368. *Whelan*, 797 F.2d at 1232-33. For a discussion of this opinion, see *supra* text accompanying notes 173-201.

369. For a discussion of the danger of expert opinions overwhelming factfinders, *see supra* text accompanying notes 346-348. Successive filtering may not even avoid this danger. *See id.*

ideas.[370] In the past, however, courts have taken this task upon themselves, often identifying only one idea in a program and thereby extending potentially overbroad protection to that program.[371] Because successive filtering calls for experts to carefully apply the abstractions test in order to separate out the ideas at each level of a program, its inclusion in the integrated test helps to ensure that copyright protection will only extend to the elements of a program that are truly creative expression.[372] The abstractions test recognizes that a literary work may contain any number of ideas, and careful application of that test under the integrated approach provides a principled means of identifying those ideas.[373]

The opinion in *Synercom Technology, Inc. v. University Computing Co.*,[374] discussed and criticized in Part IV above, demonstrates that courts alone are unable to comprehend the workings of software programs and the nature of their elements.[375] The *Synercom* court attempted to analyze input formats on its own by analogizing them to something more familiar to the court, namely the "figure H" pattern for a gear stick.[376] Unfortunately, analogies of this sort tend to be circular; the ultimate outcome of the case is determined by the characteristics of the familiar item, rather than those of the item at issue.[377] The *Lotus* court recognized that analogies can be misleading, if not carefully applied.[378] The integrated

---

370. *See supra* text accompanying notes 329-331 for a discussion on the importance of defining the ideas in a work properly.

371. *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1236 (3d Cir. 1986) (idea of program is its purpose), *cert. denied*, 479 U.S. 1031 (1987); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 65 (D. Mass. 1990) (idea of program is spreadsheet); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (idea of program is printing cards).

372. For an explanation of the use of the abstractions test in successive filtering, *see supra* text accompanying notes 328-331.

373. For a discussion of why the abstractions test is particularly well-suited to software infringement cases, *see supra* note 108 and accompanying text.

374. 462 F. Supp. 1003 (N.D. Tex. 1978).

375. For a discussion of the *Synercom* opinion and a critical analysis of its reasoning, *see supra* text accompanying notes 133-146.

376. *Synercom*, 462 F. Supp. at 1013.

377. *See* text accompanying notes 145-146 for a discussion of how this problem occurred in *Synercom*.

378. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 71-73 (D. Mass. 1990). The *Lotus* court had this to say about analogies: "They are not to be mistaken . . . for logically compelled inferences from authoritative declarations." *Id.* at 71.

approach avoids this danger by relying on expert testimony to identify the characteristics of each program element during successive filtering and to preserve those which contain protectable expression. By utilizing expert testimony during the successive filtering portion of the inquiry, the integrated approach allows determinations to be made based on the nature of the program elements themselves, not on the nature of whatever item the court chooses to compare them to.

Because of the large amount of material in programs taken from the public domain, it is especially important to exclude unprotectable elements before considering whether infringement occurred.[379] The court's decision in *E.F. Johnson Co. v. Uniden Corp. of America*,[380] discussed above in Part IV, demonstrates how courts abandoning the bifurcated test can erroneously base a finding of infringement at least partially on permissible similarities between unprotected program elements.[381] The *Johnson* court appears to have conducted its analysis in reverse order, finding infringement first, and then eliminating unprotectable elements from consideration.[382] By carefully eliminating all but the protected "core" of a program from consideration before the jury views it, the integrated approach will eliminate this danger.[383]

Several courts addressing copyright protection for computer programs have noted, quite properly, that external considerations should limit the extent to which certain non-literal elements should be protected. The court in *Plains Cotton Cooperative Association v. Goodpasture Computer Service*,[384] discussed in Part IV above, for example, stated that programming choices dictated by considerations of the program's subject, the cotton market, did not merit protection.[385] Also, the court in *Q-Co Industries v. Hoff-*

379. For a discussion of the "stepping stone" method of programming, by which new programs incorporate public domain elements of previously-existing programs, *see supra* text accompanying notes 74-77.

380. 623 F. Supp. 1485 (D.C. Minn. 1985).

381. For a discussion of the *Johnson* opinion, *see supra* text accompanying notes 147-164.

382. *Johnson*, 623 F. Supp. at 1493-99.

383. For a discussion of how this approach will also avoid the more subtle danger under the traditional bifurcated test that the factfinder will be influenced by similarities between unprotected elements viewed during the extrinsic branch of the test, *see supra* text accompanying notes 366-369.

384. 807 F.2d 1256 (5th Cir.), *cert. denied*, 484 U.S. 821 (1987).

385. *Plains Cotton*, 807 F.2d at 1262. For a discussion and analysis of this opinion, *see supra* text accompanying notes 219-226.

man,[386] discussed and criticized in Part IV above, stated that to the extent that certain modules would be present in any tele-prompting program, those modules could not be protected by copyright.[387] These statements are, at least implicitly, attempts to apply the same policies that underlie the merger and scenes a faire doctrines of copyright law.[388] These courts, however, have failed to conduct their analyses under traditional copyright doctrines, leading to less effective implementation of the policies behind those doctrines. For example, the *Q-Co* court conducted no analysis of the issue of whether the plaintiff's program modules contained any unique expression, or whether all of the elements of those modules would in fact exist in every tele-prompting program. Instead, the court noted that the plaintiff had not introduced any expert testimony on the issue.[389] Thus, although the *Q-Co* court recognized that external considerations could limit copyright protection in certain instances, it failed to evaluate the significance of such external considerations in the case before it, as required by both merger and scenes a faire; instead, the court placed that burden on the plaintiff. The integrated approach places the burden on the court to identify and consider the protected elements of each work, rather than forcing the plaintiff to justify them through expert testimony. By carefully adhering to traditional copyright doctrines and using expert assistance, the integrated approach isolates the protectable expression in a software program, rather than speculating on which elements are protectable.

Even courts which recognize traditional copyright doctrines may have difficulty applying them to computer programs, because courts lack a proper understanding of the practical realities of programming. The opinion in *Broderbund Software, Inc. v. Unison World, Inc.*,[390] discussed and criticized above, demonstrates this difficulty.[391] The *Broderbund* court rejected the defendant's argu-

386. 625 F. Supp. 608 (S.D.N.Y. 1985).

387. *Q-Co*, 625 F. Supp. at 616. This opinion is discussed *supra* text accompanying notes 165-172.

388. Merger and scenes a faire both operate to limit copyright protection in situations where the choices available to the creator are limited, thereby preventing the extension of copyright protection to ideas. For a discussion of these doctrines, *see supra* text accompanying notes 110-114, 123-125.

389. *Q-Co*, 625 F. Supp. at 616.

390. 648 F. Supp. 1127 (N.D. Cal. 1986).

391. For a discussion of this opinion, *see supra* text accompanying notes 202-218.

ment that merger should apply, stating that the fact that another program already existed which performed the same tasks as the plaintiff's program in a different manner foreclosed application of the merger doctrine.[392] Although the court's holding seems logical on its face, it is not quite accurate. Under merger, the question is whether the means of expressing an idea are narrowly limited, not whether there is at least one other way to express the idea.[393] The *Broderbund* court should have determined whether there were any practical programming considerations limiting the choices available to the defendant, rather than relying on its own observation that there was another program accomplishing the same task in a different way.[394] By employing expert testimony during its successive filtering process, the integrated approach ensures that courts will conduct the proper factual inquiry when applying copyright doctrines in software infringement cases. Although courts are well-versed in these doctrines, they are generally unfamiliar with the process of writing computer programs.[395] The integrated approach recognizes this shortcoming and employs expert testimony to "flesh out" the factual situation for courts, enabling them to properly apply the appropriate legal doctrines to the case.

The court's opinion in *Lotus Development Corp. v. Paperback Software International*,[396] discussed and criticized in Part IV above, demonstrates yet another problem encountered by courts, that of departing from well-settled legal doctrines.[397] The *Lotus* court developed a three-part test that is basically in line with the policies of copyright law.[398] Each of the elements of the court's test, however, deviates slightly from the law's standards, thereby creating requirements that are potentially at odds with the Copyright Act and its judicial gloss.[399] By carefully adhering to well-settled principles during successive filtering, the integrated ap-

---

392. *Broderbund*, 648 F. Supp. at 1132.

393. *See supra* text accompanying note 217.

394. Incidentally, the other program was "Stickybear Printer," a print-shop program designed for children. Thus, an entirely different set of considerations may have led to its different structure. *See Broderbund*, 648 F. Supp. at 1132.

395. "The challenge to counsel to make comprehensible for the court the esoterica of bytes and modules is daunting." *Q-Co*, 608 F. Supp. at 610.

396. 740 F. Supp. 37 (D. Mass. 1990).

397. The *Lotus* opinion is discussed *supra* text accompanying notes 265-288.

398. *Lotus*, 740 F. Supp. at 60-62. For a discussion of this test and how it echoes basic copyright policies, *see supra* text accompanying notes 283-288.

399. *See supra* text accompanying notes 183-188 for an explanation of this problem.

proach carefully tracks copyright law, eliminating the danger that courts may set different standards for protection in software infringement cases.

The integrated approach avoids the problem created by the *Johnson* court's "iterative" test of limiting copyright protection for non-literal program elements merely because those elements are difficult to analyze.[400] Similarly, the test avoids the approach taken by the *Whelan* court of abandoning the bifurcated test and sacrificing all of its advantages merely because that test is difficult to apply.[401] Instead, the integrated approach carefully preserves the advantages of both the traditional bifurcated substantial similarity test and the successive filtering approach and minimizes the problems that arise when applying those tests to software infringement cases.

VIII. CONCLUSION

The computer industry nets billions of dollars every year on new programs; it also loses approximately two billion dollars every year on the illegal copying of such programs. In order to achieve success in this vast consumer market, programmers must invest substantial effort in designing the non-literal elements of new programs. Indeed, these non-literal elements are the most crucial in determining the marketability of a new program. Copyright law has emerged as the principal source of protection for computer programs, but the extent of protection offered by copyright law to the non-literal elements of programs is unclear. Because such non-literal elements are expensive to develop and easy to copy, it is essential to offer protection in order to encourage innovation in this area. Without protection, the incentive to develop new user interfaces and more efficient or creative screen displays will be removed. Currently, there is uncertainty in the software industry about the extent to which such program elements can be protected.

Courts addressing the protection of non-literal program elements fear that giving protection in instances where programming choices are limited or dictated by functional or external considerations will give monopolies that would inhibit competition. Traditional copyright law, however, has already developed a set of doctrines, such as merger and the idea/expression dichotomy, to

400. For a discussion of the approach taken by the *Johnson* court and its reasons for doing so, *see supra* text accompanying notes 153-164.

401. *See supra* text accompanying note 181.

prevent giving broad protection in such situations. Application of the bifurcated substantial similarity test, integrating the successive filtering approach, will allow courts, with proper expert assistance, to apply these traditional copyright doctrines to computer programs without removing from the trier of fact the right to determine the ultimate question of infringement from the standpoint of the ordinary observer. This approach should result in the protection of those non-literal elements of a program which are truly expressive, while denying protection of ideas or expressions that are narrowly limited by efficiency or external considerations. The approach should lead to more principled decisionmaking and should bring more certainty to the software industry.