COPYRIGHTED SOFTWARE: SEPARATING THE PROTECTED EXPRESSION FROM UNPROTECTED IDEAS, A STARTING POINT†

J. DIANNE BRINSON*

Copyright protects only an author's expression of ideas, not the ideas themselves. Now that it is clear that copyright protects computer programs, courts must do for computer programs what courts have found difficult to do for literary and artistic works — separate ideas from expression. This article offers two principles or suggestions for beginning the process of separating a computer program's unprotected ideas from its protected expression: (1) do not confuse protected expression with program code; and (2) do not confuse a program's unprotected ideas with its function.

I. INTRODUCTION

If we want a computer to solve a particular problem, we must give the computer a "recipe" that spells out the exact steps which the computer must execute to reach that problem's solution.[1] The recipe must be written so precisely that the computer can follow the instructions, and it must be written in a form of notation (code) that the computer can mechanically comprehend.[2] The recipe that instructs the computer is called a program.[3] Programs commonly are referred to as "computer software."[4]

A computer can do only what it is instructed to do. A person must write the program that instructs the computer. In the United States, authors of computer programs may claim copyright protection for their works.[5] The exclusive rights of copyright give the creator of a computer program valuable protection against unauthorized copying and

---

* Associate Professor of Law, Georgia State University College of Law, Atlanta, Georgia; J.D., Yale Law School (1976); B.A., Duke (1973); admitted to the bar in Georgia and California.

[1] S. ALAGIC & M. ARBIB, THE DESIGN OF WELL-STRUCTURED AND CORRECT PROGRAMS 1 (1978).

[2] *Id.* Although a computer cannot think, it can execute correctly written instructions faster than a human could. *See* R. SALTMAN, COPYRIGHT IN COMPUTER-READABLE WORKS 59 (1977).

[3] S. ALAGIC & M. ARBIB, *supra* note 1, at 1.

[4] "Software", in its broadest sense, includes everything in a computer system that is not "hardware," the physical components of the computer that actually perform operations. This broader definition of software includes, in addition to programs, such related material as documentation, flow charts, and user manuals. Note, *Defining the Scope of Copyright Protection for Computer Software*, 38 STAN. L. REV. 497, 500 (1986). For a more detailed discussion of the term "software" see Keplinger, *Computer Software — Its Nature and Its Protection*, 30 EMORY L.J. 483, 484–88 (1981).

[5] *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1223 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1247 (3d Cir. 1983). *See generally* 1 M. NIMMER, NIMMER ON COPYRIGHT § 2.04[C] (1987). There is also an international trend toward copyright protection for software. Oman, *Software As Seen by the U.S. Copyright Office*, 28-IDEA 29, 33 (1987). The work *Intellectual Property Rights in an Age of Electronics and Information* (1986) [hereinafter *OTA Report*], prepared by the Office of Technology Assessment (OTA), "examines the impact of . . . advances in communication technologies on [our] intellectual property system." *Id.* at iii.

use of the work.[6] Copyright protection extends, however, only to the author's "expression" of ideas, and not to the underlying ideas used by that author. Ideas are open to everyone, and others are free to study a copyrighted work and take its ideas.[7]

Now that it is clear that software is protected by copyright, software producers are bringing copyright infringement suits to stop the unauthorized sale of alleged clones of their copyrighted programs. The defendant's position in such a case is likely to be that the defendant took merely unprotected ideas from the plaintiff's work.[8] Such a defense requires the court to separate a program's protected expression from its unprotected ideas. In literature and drama, the more traditional areas of copyright, courts have, for years, had trouble separating expression from ideas.[9] Courts now must distinguish between ideas and expression for computer programs, works written in technical code language for the purpose of instructing computers. The dividing line between idea and expression, elusive in literature, cannot easily be drawn for programs. As one federal district court judge noted recently, "computer programming . . . [is] not readily comprehended by the uninitiated. The challenge of counsel to make comprehensible for the court the esoterica of bytes and modules is daunting."[10]

The purpose of this article is to suggest a starting point for separating a program's ideas from its expression. This starting point consists of two principles. First, do not confuse protected expression with the program code. A program's protected expression is not limited to the literal code of the program. Second, do not confuse a program's unprotected ideas with its function. A program, whatever its function, contains many unprotected ideas.

These two principles, drawn from non-software copyright cases, discussed in Part II of this article, and from program design concepts, discussed in Part III of this article, are the focus of Part IV of this article. The Third Circuit's recent decision in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*,[11] one of the cases discussed in Part IV, recognizes the first principle but not the second principle. Part V hypothesizes that if courts

---

[6] Software is easily duplicated within the computer in a process that resembles the audio taping of a record. Piracy here and abroad is a major problem for the U.S. software industry. *See* OFFICE OF COMPUTERS AND BUSINESS EQUIPMENT, U.S. DEPARTMENT OF COMMERCE, A COMPETITIVE ASSESSMENT OF THE U.S. SOFTWARE INDUSTRY 51–52 (1984) [hereinafter REPORT]. With the appearance of the microcomputer and personal computers in the late 1970s, the software market increased dramatically. The distribution of mass-marketed programs for such microcomputers as the IBM PC has become a multibillion dollar industry. Note, *Copyright Protection of Computer Software*, 5 COMPUTER LAW J. 413, 416 n.10 (1985).

[7] The distinction between the protected expression and the unprotected ideas originated in Baker v. Selden, 101 U.S. 99 (1879). The present version of the federal copyright statute codifies the idea/expression dichotomy, stating in part that "in no case does copyright protection . . . extend to any idea . . . ." 17 U.S.C. § 102(b) (1982). *See generally* 3 M. NIMMER, NIMMER ON COPYRIGHT § 13.03[A] (1987); *infra* notes 52–63 and accompanying text.

[8] *E.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987). See *infra* Section IV for other cases analyzing this defense.

[9] *See infra* Section II D.

[10] Q-Co Indus. v. Hoffman, 625 F. Supp. 608, 610 (S.D.N.Y. 1985).

[11] 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

Whelan is the subject of a special forum issue of the *Journal of Law and Technology*. That issue contains four commentaries on *Whelan*: Ladd & Joseph, *Expanding Computer Software Protection by Limiting the Idea*, 2 J.L. & TECH. 25 (1987); Goldhammer, *Computer Programs and Technological Innovation: Testing the Copyright Law*, 2 J.L. & TECH. 17 (1987); Karam, *Countervailing Considerations*, 2 J.L. & TECH. 25 (1987); and Wessel, *Substantial Similarity*, 2 J.L. & TECH. 35 (1987).

perpetuate the *Whelan* court's error of extending copyright protection beyond program code, without recognizing that a program contains many unprotected ideas, the result will be that courts will over-protect early developers of software at the expense of later developers. Finally, after discussing the potential impact of the *Whelan* error, Part V of this article proposes that courts avoid the overprotection problem by recognizing this article's second principle that there are many unprotected ideas subsumed in a computer program's function and by using copyright's traditional "levels of abstraction" analysis.

II. The Law of Copyright

A. *Copyrightability*

Copyright law in this country is governed by a federal statute, the Copyright Act of 1976.[12] Copyright protection arises automatically when an author fixes an original work in any tangible medium of expression.[13] Under the statute the owner of a copyright may register his or her copyright claim with the Copyright Office, but no registration is required to create the rights of copyright.[14] Registration is, however, a prerequisite to initiating an action for copyright infringement.[15]

Although copyright protects only original works, an author meets the originality requirement by simply creating the work. Originality requires neither novelty nor uniqueness.[16] To be original, a work need only be created independently by the author

---

[12] Pub. L. No. 94-553, 90 Stat. 2541 (1976) (codified at 17 U.S.C. §§ 101–810 (1982)). Article I of the U.S. Constitution gives Congress the authority to enact laws to protect the rights of authors and inventors. U.S. Const. art. 1, § 8, cl.8. Congress enacted the first federal copyright statute in 1790. Act of May 31, 1790, 13th Cong., 2d Sess., 1 Stat. 124 (1790). This early statute was a forerunner of the present copyright statute, which is cited above in this footnote. The federal statute preempts any state attempts at legislating in the copyright area.

[13] 17 U.S.C. § 102(a) (1982).

[14] *Id.* §§ 102(a), 408. Sections 408–412 of the copyright statute describe the registration procedures. Briefly, one registers a copyright claim by sending an application to the Copyright Office together with two copies of the work and a ten dollar filing fee. *Id.* §§ 408, 409, 708.

[15] *Id.* §§ 411, 501(b). Early registration offers a procedural benefit. In judicial proceedings a certificate of registration made within five years of the first publication of the work is prima facie evidence of the copyright's validity. *Id.* § 410(c). The copyright statute defines "publication" as "the distribution of copies . . . of a work to the public by sale or other transfer of ownership, or by rental, lease, or lending." *Id.* § 101. Also, statutory damages or attorney's fees are not available for (1) infringement of copyright in an unpublished work before the registration date; or (2) for infringement begun between the first publication date and the registration date, if registration is not filed within three months of the first publication. *Id.* § 412. Sections 504 and 505 of the copyright statute cover damages and attorney's fees. *See id.* §§ 504, 505.

The copyright statute requires that a notice of copyright be placed on all publicly distributed copies of a work claimed to be protected by copyright in order to give reasonable notice of that claim. *Id.* § 401. There are some exceptions to that rule. The omission of notice does not invalidate a copyright if the notice was omitted from only a relatively small number of copies. *Id.* § 405(a)(1). Additionally, the omission of notice from a large number of publicly distributed copies does not invalidate the copyright if the copyright is registered within five years of the "no-notice" publication and a reasonable effort is made to add notice to all copies that were distributed in the United States. *Id.* § 405(a)(2).

[16] Alfred Bell & Co. v. Catalda Fine Arts, 191 F.2d 99, 102–03 (2d Cir. 1951). The Copyright Office does not substantively examine a work for originality before registering it. Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, 143–44 (D.N.J. 1982).

Uniqueness *is* a prerequisite for patentability. To qualify for a patent a product or process must

rather than copied from some other work. The work must owe its origin to the author claiming copyright rights.[17] Accordingly, the statute requires that the work demonstrate only minimal creativity. No artistic merit is required.[18]

The copyright statute confers on the copyright owner five exclusive rights over the copyrighted work.[19] The three rights that apply to software are the right to reproduce or copy the work; the right to distribute copies of the work to the public by sale, rental, lease, or loan; and the right to prepare derivative works based on the copyrighted work.[20] These exclusive rights are really product control rights; without them, an author would have no recourse against use and exploitation of his or her work by others.[21] One who violates any of the copyright owner's exclusive rights is an "infringer,"[22] who is liable to the copyright owner for damages and profits stemming from the infringement.[23] The federal district courts have exclusive original subject matter jurisdiction over copyright

---

be new, useful, and non-obvious to one with ordinary skill in the relevant discipline. 35 U.S.C. §§ 101–03 (1982).

[17] *Alfred Bell*, 191 F.2d at 102.

[18] *See* Bleistein v. Donaldson Lithographing Co., 188 U.S. 239, 251 (1903). *See generally* 1 M. NIMMER, *supra* note 5, §§ 2.01[B], 2.08[B].

[19] 17 U.S.C. § 106 (1982). The "owner" is normally the author, but the author may transfer all or part of the copyright and any of the exclusive rights. *Id.* §§ 201(a), (d). The rights in most works created "on the job" are held by the creator's employer, for in the case of a "work made for hire" the employer is deemed the author unless the actual creator and the employer have expressly agreed otherwise in a signed written document. *Id.* § 201(b). Section 101 of the copyright act defines "work for hire." *See id.* § 101.

[20] *Id.* § 106. The other two rights are the public performance right and the public display right. *Id.* The statute defines a "derivative work" as "a work based upon one or more pre-existing works, such as a translation . . . abridgment, condensation, [or some other transformed presentation]." *Id.* § 101. A derivative work is copyrightable separately from the base work if it meets the minimal "originality" requirement for copyright. So long as the new work contains any distinguishable variation from the base work it will have sufficient originality to support a separate copyright. 1 M. NIMMER, *supra* note 5, § 3.01. The derivative work's copyright covers only the material added to the underlying work to create the derivative work. 17 U.S.C. § 103(b) (1982).

[21] *See* 17 U.S.C. § 301 (1982) and 35 U.S.C. § 6281 (1982). Copyright, patent, and trademark law sometimes are referred to as "intellectual property" law. *E.g.*, EPSTEIN, MODERN INTELLECTUAL PROPERTY (1986); KINTER & LAHR, AN INTELLECTUAL PROPERTY PRIMER (1975). This body of law enables the creator of some product of mental effort to exercise ownership control rights somewhat similar to rights given by law to the owner of a piece of real or personal property.

[22] 17 U.S.C. § 501(a) (1982). The copyright owner's exclusive rights are not absolute. Those rights are subject to "fair use" by others, which can be loosely characterized as reasonable limited use of the work, for purposes such as scholarship or literary criticism. The fair use doctrine was first developed by the courts but is now codified. *See id.* § 107. Section 107 sets out four factors for courts to consider in determining whether an unauthorized use is "fair use". *Id. See generally* Note, *Toward a Unified Theory Of Copyright Infringement For an Advanced Technological Era*, 96 HARV. L. REV. 450 (1982). Sections 108–18 of the Copyright Act add further details concerning particular uses of certain types of copyrighted works by non-owners. *See* 17 U.S.C. §§ 108–18 (1982). Recently, the Court of Appeals for the Eleventh Circuit considered the limits of "fair use" in Pacific and Southern Co. v. Duncan, 744 F.2d 1490, 1494–95 (11th Cir. 1984), *cert. denied*, 471 U.S. 1004 (1985).

The "first sale" doctrine places a further limitation on the copyright owner's control rights. *See* 17 U.S.C. § 109 (1982). Under the "first sale" doctrine the owner of a copy of a copyrighted work has the right to sell or dispose of that copy without the permission of the copyright owner. *Id.* Ownership of the copyright is distinct from ownership of the material object (book, diskette, etc.) in which the work is embodied. *Id.* § 202.

[23] *Id.* §§ 504, 505.

infringement actions[24] and may grant temporary and final injunctions to prevent or restrain acts of infringement.[25] Such injunctions are operative throughout the United States.[26]

B. *Copyrightability of Computer Programs*

The Copyright Office began registering copyrights for computer programs in 1964.[27] The legislative history of the 1976 Copyright Act indicates that Congress intended for the revised copyright statute to protect computer programs. The House Committee Report stated that copyrightable literary works[28] "include computer data bases, and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves."[29] Congress resolved any remaining doubts as to copyrightability of computer programs in December, 1980, by enacting the Computer Software Copyright Act of 1980, an amendment to the copyright statute.[30] The Computer Software Copyright Act, according to its legislative history, clearly applied the 1976 copyright law to computer programs.[31] The 1980 Act added a definition of "computer program" to the definitions section of the copyright

---

[24] 28 U.S.C. § 1338 (1982).

[25] 17 U.S.C. § 502(a) (1982).

[26] *Id.* § 502(b).

[27] One commentator has written that the Copyright Office had both statutory doubt and constitutional doubt concerning the copyrightability of computer programs. However, the Copyright Office began permitting registration of programs, leaving a determination of copyrightability to the courts. Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-readable Form*, 1984 DUKE L.J. 663, 692 n.109. The "statutory doubt" was whether machine-readable versions of programs could be "copies" within the meaning of that term as it was used in the 1909 copyright statute, the immediate predecessor to the present statute. *Id.* at 692 n.109. In White-Smith Music Publishing Co. v. Apollo Co., the Supreme Court held that copyright protection extended only to "copies" which were perceptible to humans. 209 U.S. 1, 17 (1908). The doubt on the constitutional level was whether machine-readable programs could be considered an author's writings. Samuelson, *supra*, at 692 n.109. The enabling clause in the Constitution authorizes Congress to enact legislation giving authors rights to their writings. U.S. CONST. art. 1, § 8, cl.8. In Data Cash Systems, Inc. v. JS & A Group, Inc., 480 F. Supp. 1063, 1069 (N.D. Ill. 1979), *aff'd on other grounds*, 628 F.2d 1038 (7th Cir. 1980), decided under the 1909 statute, the court concluded, based on *White-Smith*, that a program in its object code phase in the form of a ROM installed in a computer's circuitry was a machine part rather than a copy of the source code program. See *infra* part III A for a discussion of these technical terms. *See* Note, *Computer Copyright Law: An Emerging Form of Protection For Object Code Software After Apple v. Franklin*, 5 COMPUTER LAW J. 233, 243–46 (1984). The current statute avoids the *White-Smith* problem, for it provides for copyright protection for "original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, *either directly or with the aid of a machine or device*." 17 U.S.C. § 102(a) (1982) (emphasis added).

[28] 17 U.S.C. § 102(a)(1) (1982).

[29] H.R. REP. No. 1476, 94th Cong., 2d. Sess. 51, 54 (1976) *reprinted at* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5667 [hereinafter H.R. REP. No. 1476]. The Senate Report on the 1976 Act was somewhat ambiguous on the question of whether computer programs were "literary works". See 1 M. NIMMER, *supra* note 5, § 2.04[C], at 2-43 & n.21. For more on the attention given to copyrightability of programs in drafting the 1976 Act, see Keplinger, *supra* note 4.

[30] Act of December 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028–29.

[31] H.R. REP. No. 1307, 96th Cong., 2d. Sess., *reprinted at* 1980 U.S. CODE CONG. & ADMIN. NEWS 6460, 6486 [hereinafter H.R. REP. No. 1307]; *see also* 1 M. NIMMER, *supra* note 5, § 2.04[C], at 2-44.

statute.[32] Furthermore, the revised copyright statute now makes it clear, by implication, that unauthorized reproductions and adaptations of computer programs are infringing acts unless the reproduction or adaptation falls within the scope of the statute's express grant of the right to make copies and adaptations of copyrighted programs for specified purposes.[33]

For the last several years, courts have routinely held that computer programs are copyrightable "works of authorship."[34] In the software infringement cases decided to date, courts have granted relief for infringement of the exclusive copying, distribution, and derivative works rights.[35] The Copyright Office now routinely registers copyrights for software, and software developers routinely file applications to register programs.[36]

---

[32] 17 U.S.C. § 101 (1982). Section 101 now provides, in part, that a "computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." *Id.*

[33] *Id.* § 117. The owner of a copy of a program may copy the program for archival purposes and when necessary for use of the program in the computer. *Id.*

These 1980 amendments were recommended by the Commission on New Technological Uses of Copyrighted Works (CONTU). H.R. REP. No. 1307, *supra* note 31, at 6482. Congress created CONTU in 1974 to study copyright problems raised by growth of the computer industry. Act of Dec. 31, 1974, Pub. L. No. 93-573, §§ 201–08, 88 Stat. 1873, 1873–75. CONTU recommended that Congress amend the 1976 copyright act "to make it explicit that computer programs, to the extent they embody an author's original creation, are proper subject matter of copyright." NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT, (July 31, 1978) [hereinafter CONTU REPORT]. CONTU Commissioner John Hersey argued against extending copyright protection to programs. *Id.* at 69. For a detailed discussion of CONTU's work, see Keplinger, *supra* note 4; Samuelson, *supra* note 27.

[34] *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987); Apple Computer, Inc., v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983); Williams Elecs., Inc. v. Artic Int'l., Inc., 685 F.2d 870 (3d Cir. 1982).

[35] *See, e.g.*, *Whelan*, 797 F.2d at 1248 (copying and distribution); SAS Institute, Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816 (M.D. Tenn. 1985) (derivative work).

[36] Programs are registered as "literary works." 17 U.S.C. § 102(a)(1) (1982). Some authors also have sought to register screen displays as "audiovisual works." At the moment the Copyright Office's position is that the audiovisual registration is superfluous. That viewpoint recently was called into question by Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449 (N.D. Ga. 1987), where the court held that the copyright on a computer program does not extend to screen display. *Id.* at 456. *Contra* M. Kramer Mfg. Co. v. Andrews, 783 F.2d 421, 422 (4th Cir. 1986), *cert. denied*, 107 S. Ct. 77 (1987) (copyright on the visual display protects the underlying program that produces the screen display).

In September, 1987, the Copyright Office held a hearing on the question of whether it should allow separate registration of screen displays. Mace, *Apple Urges Separate Copyright Registration of Screen Displays*, INFOWORLD 101 (Sept. 14, 1987). In June of 1988, the Copyright Office issued a registration decision stating that, in its view, all copyrightable expression embodied in a program, including screen displays, is a single work and should be registered on a single application form. Notice of Registration Decision: Registration and Deposit of Computer Screen Displays, 53 Fed. Reg. 21817 (1988) (37 CFR Pt. 202). Audiovisual copyrights *are* available for videogames, which are operated by computer programs.

The writing on the subject of copyright protection for software has mushroomed in the past few years. On the practical level, see, *e.g.*, T. SMEDINGHOFF, THE LEGAL GUIDE TO DEVELOPING, PROTECTING, AND MARKETING SOFTWARE (1986). On the scholarly level, see Conley & Bryan, *A Unifying Theory For the Litigation of Computer Software Copyright Cases*, 63 N.C.L. REV. 563 (1985); Karjala, *Copyright, Computer Software, and The New Protectionism*, 28 JURIMETRICS 33 (1987); Keplinger, *supra* note 4; Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329 (1987); Nimmer & Krauthaus, *Copyright and Software Technology Infringement: Defining Third Party Development*

Many software distributors also attempt to use state trade secrets laws to protect software from piracy.[37] Patent protection is also available for software that meets the stringent requirements for a patent grant.[38]

C. *An Infringement Case*

In a copyright infringement action the plaintiff makes out his or her prima facie case by showing that the plaintiff owns a valid copyright; and that the defendant "copied" the work protected by the plaintiff's copyright.[39] A plaintiff's copyright is valid if the work claimed to be protected is an original work of authorship and is appropriate copyright subject matter.[40] As to the "copying" element, the plaintiff rarely can obtain direct evidence that the defendant copied the work.[41] Rather, the plaintiff generally puts in circumstantial evidence of the defendant's copying. This circumstantial evidence consists of two elements. First, the plaintiff shows that the defendant had access to the infringed work.[42] To date plaintiffs in software copyright cases have been able to prove

---

Rights, 62 IND. L.J. 13 (1986); Samuelson, *supra* note 27; Note, *supra* note 4; and Note, *supra* note 27. The University of Pittsburgh Law Review devoted an entire issue to the future of software protection (Vol. 47, No. 4). That issue includes Davidson, *Common Law, Uncommon Software*, 47 U. PITT. L. REV. 1037 (1986) and Goldstein, *Infringement of Copyright in Computer Programs*, 47 U. PITT. L. REV. 1119 (1986).

[37] Many writers have noted that state law trade secret protection may be used to supplement copyright protection on software. *See, e.g.*, T. SMEDINGHOFF, *supra* note 36; Conley & Bryan, *supra* note 36; Gilburne & Johnston, *Trade Secret Protection for Software Generally and in the Mass Market*, 3 COMPUTER L. J. 211 (1982); Yoches, *Protection of Computer Software by Patents, Trade Secrets, and Trademarks*, 22 TORT AND INSUR. L.J. 354 (1987); Note, *supra* note 6; Note, *supra* note 4.

For an overview of international protection for software, see Nimmer & Krauthaus, *Classification of Computer Software For Legal Protection: International Perspectives*, 21 INT'L L. 733 (1987).

[38] Patent protection is available only for non-obvious works. 35 U.S.C. § 103 (1982). One commentator has stated that very few programs could meet the non-obviousness requirement. *See* I.D. BENDER, COMPUTER LAW § 3A.02 (1984). Another writer, however, considers patent protection of software to be of growing importance. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611, 648. *See* Dann v. Johnston, 425 U.S. 219, 229 (1976) (program not patentable because it was obvious). Furthermore, a patent will be denied if it is determined that what is offered as a patentable process is really an unpatentable mathematical algorithm. *See* Gottschalk v. Benson, 409 U.S. 63 (1972). *See generally* Davidson, *supra*, at 634–51.

[39] *See generally* 3 M. NIMMER, *supra* note 5, § 13.01.

[40] 17 U.S.C. §§ 102–03 (1982). A copyright registration certificate issued within five years of the first "publication" of the work is prima facie evidence of the copyright's validity. *Id.* § 410(c). Once the plaintiff establishes prima facie proof of validity through the registration certificate, the burden of persuasion as to the invalidity of the copyright is on the defendant. 3 M. NIMMER, *supra* note 5, § 12.11.

To prove ownership, a plaintiff who authored the work in issue need only prove authorship. If, however, the plaintiff acquired the copyright by transfer or assignment, and is not the party to whom the copyright was registered, the plaintiff must prove chain of title. If the non-author plaintiff took an assignment of the copyright from the original owner and then registered the copyright in his or her own name, the certificate of registration is prima facie evidence of the ownership. *See* 3 M. NIMMER, *supra* note 5, § 12.11.

[41] Defendants may, however, admit to copying the work, obviating the need to prove copying. *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1245 (3d Cir. 1983) (defendant admitted copying the work, but raised various defenses, including absolute non-copyrightability of software).

[42] *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1232 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987). *See generally* 3 M. NIMMER, *supra* note 5, § 12.11[D]. If the

the defendant's access to the work.[43] Second, the plaintiff shows that the defendant's work is "substantially similar" to the allegedly infringed work.[44]

The well-known copyright treatise by Nimmer notes that determining what constitutes substantial similarity, and thus infringes a copyright, is "one of the most difficult questions in copyright law, and one that is the least susceptible of helpful generalization."[45] At one extreme, two works need not be literally identical to be substantially similar for purposes of copyright infringement.[46] Thus, a defendant cannot escape a finding of substantial similarity by paraphrasing rather than repeating the plaintiff's work.[47] At the other extreme, if the only similarity between the plaintiff's and defendant's work is the use of the same abstract ideas, substantial similarity is lacking,[48] for copyright does not prohibit persons from borrowing abstract ideas contained in a copyrighted work.[49] For a court to find infringement, the defendants must have copied not just the plaintiff's ideas, but the plaintiff's expression of the ideas.[50] Accordingly, the substantial similarity that constitutes copyright infringement must be expression-level similarity, not merely idea-level similarity.[51]

---

plaintiff cannot prove access by direct evidence, he or she may instead show that the similarity between the defendant's work and the allegedly infringed work is so striking that the court may infer access. Arnstein v. Porter, 154 F.2d 464, 468 (2d Cir. 1946).

[43] *See, e.g., Whelan*, 797 F.2d at 1232 (the defendant had used and marketed the plaintiff's program); Digital Communications Assocs. v. Softklone Distrib. Co., 659 F. Supp. 449, 464–65 (N.D. Ga. 1987) (the defendants acknowledged access to the plaintiff's mass-marketed program).

[44] *Whelan*, 797 F.2d at 1232. Proof of access plus substantial similarity permits but does not require a finding of copying. The trier of fact may believe that the defendant independently created the work without copying the plaintiff's copyrighted work. 3 M. NIMMER, *supra* note 5, § 12.11[D], at 12-83. An independently created work that is identical to a pre-existing work does not violate the copyright on the earlier work. Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54 (2d Cir.), *cert. denied*, 298 U.S. 669 (1936).

Patent, unlike copyright, does protect against independently created duplication. Alfred Bell & Co. v. Catalda Fine Arts, Inc., 191 F.2d 99, 103 (2d Cir. 1951).

[45] 3 M. NIMMER, *supra* note 5, § 13.03, at 13-20. In the most recent of the software copyright cases, the defendant has denied that the defendant's work is substantially similar to the plaintiff's. See *infra* Part IV.

[46] 3 M. NIMMER, *supra* note 5, at § 13.03, at 13-20 & n.4; *see, e.g.*, Sid & Marty Krofft Television Prod., Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977) (defendants' television commercials infringed the copyright on plaintiff's television show for children).

[47] 3 M. NIMMER, *supra* note 5, § 13.03, at 13-20.1, 13-20.2, & n.8.

[48] *Id.* § 13.03[A], at 13-21.

[49] 17 U.S.C. § 102(b) (1982) (codification of earlier case law establishing the principle of the non-protectability of ideas).

[50] 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-21. The topic of proving substantial similarity in software copyright cases has been the focus of three recent law review articles. *See* Conley & Bryan, *supra* note 36; Comment, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test*, 68 MINN. L. REV. 1264 (1984); Note, *supra* note 4.

[51] One may ask the two relevant questions in either order: (1) Is the defendant's work as a whole substantially similar to the plaintiff's work? If yes, is the similarity more than idea-level similarity? or (2) Does any expression contained in the plaintiff's work appear in the defendant's work? If yes, was sufficient expression taken so that the defendant's work is substantially similar to the plaintiff's, justifying a finding of infringement? Whatever the order of the questions, the court ultimately must separate expression from idea. *See infra* part II D.

The authors Conley and Bryan contend that liability in a software copyright case should not depend on a showing of substantial similarity between the plaintiff's and defendant's works. Rather, they argue that the focus in software copyright litigation should be the defendant's conduct. Conley & Bryan, *supra* note 36, at 608–13.

D. Separating Ideas from Expression: The General Framework of the Analysis of the Dichotomy

Copyright does not protect the copyright owner from having others take the ideas used in the copyrighted work.[52] This "axiom of copyright law"[53] has long been recognized by the courts[54] and now is stated expressly in the copyright statute.[55] The rationale underlying this axiom is the policy underlying copyright grants: The purpose of copyright is to promote learning and progress in intellectual pursuits by encouraging authors to make their works available to the public.[56] Copyright protection for abstract ideas would undermine that goal by permitting one author to withdraw ideas from the pool of materials available to other authors.[57] Copyright thus must balance two interests: protecting authors' efforts and keeping ideas available for others to use.[58]

The 1879 case of *Baker v. Selden*[59] laid the foundation for the idea/expression dichotomy. In *Baker*, the plaintiff sought protection for bookkeeping forms contained in a book that explained a novel method of bookkeeping.[60] The Supreme Court held in *Baker* that the copyright on a book does not give the copyright holder "an exclusive property in the art described therein."[61] Furthermore, the Court ruled, where use of the idea necessarily requires copying of the work itself, that copying is not infringement.[62] Accordingly, an author may not use the copyright laws to obtain a monopoly on a system or method. In the 1954 case of *Mazer v. Stein*, the Supreme Court interpreted *Baker* to hold that copying an idea without copying the expression does not constitute infringement.[63]

Because copyright does not protect against the taking of ideas, the "substantial similarity" that serves as indirect proof of copying must be expression-level similarity, not merely idea-level similarity.[64] Over the years, many courts have noted that separating

---

[52] 17 U.S.C. § 102(b) (1982).

[53] Many cases refer to this principle as an axiom of copyright law. *See, e.g.*, Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1163 (9th Cir. 1977). *See generally* 3 M. NIMMER, *supra* note 5, § 13.03.

[54] *See, e.g.*, Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54 (2d Cir.), *cert. denied*, 298 U.S. 669 (1936) (others may copy the theme or idea of a work, but not its expression); Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

[55] 17 U.S.C. § 102(b) (1982). The legislative history indicates that codification of the idea/expression dichotomy "in no way enlarges or contracts the scope of copyright protection under the present law. Its purpose is to restate, in the context of the new single Federal system of copyright, that the basic dichotomy between expression and idea remains unchanged." H.REP. No. 1476, *supra* note 29, at 5670.

[56] *See* Chafee, *Reflections on the Law of Copyright*: I, 45 COLUM. L. REV. 503, 504–15 (1945).

[57] Use of the unprotected ideas sometimes erroneously has been called "fair use". *See, e.g.*, *Sheldon*, 81 F.2d at 54. See *supra* note 22 for a discussion of the "term of art" meaning of "fair use."

[58] Sid & Marty Krofft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157, 1163 (9th Cir. 1977).

[59] 101 U.S. 99 (1879).

[60] *Id.* at 100.

[61] *Id.* at 102.

[62] *Id.* at 103. See 1 M. NIMMER, *supra* note 5, § 2.18 for a discussion of other interpretations of *Baker* (blank forms are not copyrightable; copying for use is not infringement).

[63] 347 U.S. 201, 217 (1954). For a modern-day case similar to *Baker*, see Universal Athletic Sales Co. v. Salkeld, 511 F.2d 904 (3d Cir. 1975), where the court denied copyright protection for the exercises shown in plaintiff's work.

[64] Sid & Marty Krofft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157, 1164 (9th Cir. 1977).

a work's ideas from its protected expression is difficult.[65] Of course, if the copyright defendant literally duplicates the copyrighted work — as for example, a word-for-word appropriation of an entire song or poem — the defendant clearly has taken the author's expression.[66] A defendant may take protected expression, however, without engaging in word-for-word copying. A playwright may pirate an earlier play's dramatic meaning, for example, and violate its copyright, without copying the dialogue itself.[67] Neither product duplication nor near-identity is necessary to establish infringement.[68] The defendant's work will have expression-level similarity with the plaintiff's if the "total concept and feel" of the two works is the same.[69]

Infringement is not confined to exact repetition of the copyrighted work in its original form, but also includes other modes of adaptation, imitation, transfer or reproduction, in which the defendant may have altered the copyrighted work to disguise the piracy.[70] A defendant may not escape liability by showing that the new work is in a different medium from the copyrighted work,[71] for the copyright rights apply whatever the medium.[72] Adding new material to that taken does not create immunity,[73] nor does changing details.[74]

Although some courts have stated rules or generalizations concerning the separation of ideas from expression,[75] courts more often recognize that separating elements of the

---

[65] *See, e.g.*, Herbert Rosenthal Jewelry v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971) (quoting Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) ("no principle can be stated as to when an imitator has gone beyond copying the 'idea', and has borrowed its 'expression'")).

[66] Literal similarity — virtual word-for- word taking — is always a similarity as to the expression. 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-35.

[67] Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 55 (2d Cir. 1936) ("A play may be pirated without using the dialogue," for "speech is only a small part of a dramatist's means of expression."). See *infra*, Part IV, for a discussion of this principle's application to software.

[68] *Sid & Marty Krofft*, 562 F.2d at 1167.

[69] *See id. See generally* 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-30 & n.36.

[70] Universal Pictures Co., Inc. v. Harold Lloyd Corp., 162 F.2d 354, 360 (9th Cir. 1947).

[71] *See, e.g.*, Twentieth Century-Fox Film Corp. v. MCA, Inc., 715 F.2d 1327, 1328, 1329 n.4 (9th Cir. 1983) (the copyrighted works were a book and motion picture, whereas the defendant produced a motion picture and a television show); Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, 139, 147 n.22 (D.N.J. 1982) (copyrighted work was an arcade-size videogame, whereas defendant's game was hand-size).

[72] *See generally* 2 M. NIMMER, *supra* note 5, § 8.01[B] (1987). In such cases the defendant's work may be a derivative work based on the copyrighted work rather than a copy of that work. Although a copyright owner has the exclusive right to make derivative works, giving too expansive a definition to the concept of "derivative works" will create a conflict with the axiom that ideas are not protected. Conley & Bryan, *supra* note 36, at 572–73.

[73] Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 56 (2d Cir. 1936) ("No plagiarist can excuse the wrong by showing how much of his work he did not pirate."). The defendants in *Sheldon* maintained that much of their movie owed nothing to the plaintiff's copyrighted play. *Id.* Judge Hand agreed, but, nonetheless, found infringement. *Id. See generally* 1 M. NIMMER, *supra* note 5, § 3.03.

[74] For a modern-day application of this principle, see the recent videogame case Atari, Inc. v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 619 (7th Cir.), *cert. denied*, 459 U.S. 880 (1982).

[75] *E.g.*, Shipman v. R.K.O. Radio Pictures, Inc., 100 F.2d 533, 537 (2d Cir. 1938) (play's theme is an unprotected idea); *see, e.g.*, Reyher v. Childrens' Television Workshop, 533 F.2d 87, 91 (2d Cir. 1976) ("infringement lies not in taking a general theme but in taking its particular expression through similarities of treatment, details, scenes, events and characterization"); Dymow v. Bolton,

plaintiff's work into protected expression and unprotected idea must be undertaken "ad hoc."[76] Many courts, after identifying the common elements in the plaintiff's and defendant's works, draw upon guidance offered by Judge Learned Hand, Professor Zachariah Chafee, or Professor Nimmer.[77] These three scholars have suggested ways to characterize the common elements in copyrighted works. In the 1930 case of *Nichols v. Universal Pictures Corp.*, Judge Hand formulated his "abstractions test," which provides a framework for examining the plaintiff's work:

> Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in these series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended.[78]

The abstractions test does not tell the factfinder where to draw the line between ideas and expression. Instead, the "test" gives the trier of fact an analytical structure to use in comparing the plaintiff's and defendant's works.

Professor Chafee, writing in 1945, offered further guidance about line-drawing, stating that "the protection covers the 'pattern' of the work."[79] Professor Nimmer combined Hand's observation that any work can be broken into patterns at different levels of abstraction with Chafee's suggestion that substantial similarity should be determined by comparing common elements at a level that is somewhat abstract but still concrete enough to constitute an expression.[80] Levels of abstraction provide answers to the question "What is the work about?" The most abstract answer might be, to use *Romeo and Juliet* as an example, "boy meets girl." A slightly less abstract answer is "boy meets girl, and the two come from hostile groups." An even less abstract answer is "boy meets girl at a dance, the two come from hostile groups, and they eventually take the marriage vows." The least abstract answer is the exact language chosen by the author Shakespeare to express that theme of "boy meets girl." Nimmer's treatise uses a comparison of "Romeo and Juliet" and "West Side Story" to illustrate this approach to the idea/expression distinction, identifying thirteen common elements and concluding that the common elements form a pattern sufficiently concrete to provide a basis for a finding that the two works are substantially similar.[81] The pattern of the works are similar, involving a

---

11 F.2d 690, 692 (2d Cir. 1926) (sometimes cited as holding that plot is not copyrightable, *but see* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930), disputing that point).

[76] *See, e.g.*, Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960).

[77] *See, e.g.*, Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1163–64 (9th Cir. 1977) (citing Hand and Chafee).

[78] *Nichols*, 45 F.2d at 121 (citations omitted).

[79] Chafee, *supra* note 56, at 513. Chafee gives as an example the idea of an Irish-Jewish marriage. That idea may be borrowed from a play and some characters and situations inevitably go with that idea and may be borrowed. The play's pattern — the sequence of events and the development of the interplay of the characters — must not be followed scene by scene. *Id.* at 513–14. Chafee's example is drawn from *Nichols*, 45 F.2d 119.

[80] 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-22–13-23.

[81] *Id.* at 13-25–13-27. *But see* Knowles & Palmieri, *Dissecting Krofft: An Expression of New Ideas in Copyright?*, 8 SAN FERN. V.L. REV. 109, 137 (1980) (criticizing Nimmer's conclusion).

meeting and romance between a boy and girl from hostile groups, with resulting killings and misunderstandings.

Under certain circumstances, a second author is free to copy even expression. If the author's idea may only be expressed in more or less stereotyped form, that expression is free for the taking.[82] Otherwise, copyright protection of the expression would amount to a grant of a monopoly over the idea itself.[83] The leading case for this exception to the idea/expression dichotomy is *Herbert Rosenthal Jewelry Corp. v. Kalpakian*, decided in 1971. In *Kalpakian*, the plaintiff alleged that the defendant infringed the plaintiff's copyright on jeweled bee pins.[84] The defendant, according to the court, was free to take the "idea" of making jeweled pins in the shape of a bee. The plaintiff's protection was limited to its pin's particular design or expression. In the court's view, however, other expressions did not exist.[85] The idea and its expression appeared to be indistinguishable, noted the court.[86] Accordingly, the court ruled that the similarity between the plaintiff's and defendant's pins was inevitable, flowing from the use of the "bee pin" concept.[87] Where the idea and expression are inseparable, concluded the court, copying the expression must be permitted.[88] This principle sometimes is referred to as "merger" or "unity" of idea and expression.[89]

Later authors also are free to use incidents and plots that necessarily flow from a theme or setting.[90] These "*scenes a faire*" are indispensable or standard to the treatment of a theme, and so must be free for the taking.[91] Thus, the author of a fictionalized account of the Hindenburg's last voyage may use a German beerhall revelry scene even though an earlier work with the same theme also had a beerhall scene.[92]

Over the years, courts have separated unprotected ideas from protected expression for works of varying types, including fiction and plays,[93] non-fiction,[94] songs,[95] labels,[96] jewelry,[97] games and con-

---

[82] Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971). *See generally* 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-33 & n.44.

[83] *Kalpakian* at 742 (citing, *inter alia*, Baker v. Selden, 101 U.S. 99, 103 (1879)).

[84] *Id.* at 739.

[85] *Id.* at 742.

[86] *Id.*

[87] *Id.*

[88] *Id.* The First Circuit later expanded this principle, holding that if an idea is susceptible of "only a limited number" of expressions, no particular form is copyrightable. Morrissey v. Procter & Gamble Co., 379 F.2d 675, 678–79 (1st Cir. 1967).

[89] In considering whether an idea and its expression are indistinguishable, the analyst must state the idea abstractly rather than narrowly. Otherwise the idea *will* almost inevitably seem to merge with its expression. 3 M. NIMMER, *supra* note 5, at 13-34 n.45; Conley & Bryan, *supra* note 36, at 590.

[90] 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-32 & n.43.

[91] Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980).

[92] *Id.*

[93] *See, e.g.*, Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936); Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930).

[94] *See, e.g.*, Harper and Row Publishers, Inc. v. Nation Enters., 471 U.S. 539 (1985). *See generally* Gorman, *Copyright Protection for the Collection and Representation of Facts*, 76 HARV. L. REV. 1569 (1963).

[95] *See, e.g.*, Arnstein v. Porter, 154 F.2d 464 (2d Cir. 1946).

[96] *See, e.g.*, Kitchens of Sara Lee, Inc. v. Nifty Foods Corp., 266 F.2d 541 (2d Cir. 1959).

[97] *See, e.g.*, Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738 (9th Cir. 1971).

tests,[98] works of visual art,[99] television show characters,[100] and videogames.[101] Some commentators have stated that this idea/expression separation is really just a vehicle for a policy decision as to whether the court should rule for the plaintiff in a copyright suit.[102] These commentators argue that there is no real dichotomy of ideas and expression, because "an idea stated is an idea expressed."[103] Idea and expression are "entwined" in any given work.[104] These commentators maintain that when a court decides that some feature of a work is unprotected idea rather than protected expression, the court is merely balancing the degree of protection granted the original author against the degree of freedom allowed later authors.[105] If the unprotected "idea" is stated very abstractly — "boy meets girl" — then most of the first author's work becomes protected expression, unavailable for use by a second author.[106] Conversely, if the work is deemed to contain a number of more detailed and specific unprotected ideas, the second author's freedom is expanded and the copyright owner's protection narrowed.[107]

III. The Nature and Design of Computer Programs

Software copyright cases traditionally include some sort of "technical primer" on the nature of computers and computer programs.[108] In today's typical software copyright case, the factfinder must be able to evaluate (1) the plaintiff's contention that the defendant's program is "substantially similar" to the plaintiff's copyrighted work;[109] and (2) the defendant's contention that the defendant took, if anything, only unprotected

---

[98] *See, e.g.*, Morrissey v. Procter and Gamble Co., 379 F.2d 675 (1st Cir. 1967); *cf.* Landsberg v. Scrabble Crossword Game Players, Inc., 736 F.2d 485 (9th Cir. 1984) (author of a game strategy book claimed that the game's manufacturer infringed his copyright in writing its "player's handbook").

[99] *See, e.g.*, Franklin Mint Corp. v. National Wildlife Art Exch., Inc., 575 F.2d 62 (3d Cir. 1978) (painting); Roth Greeting Cards v. United Card Co., 429 F.2d 1106 (9th Cir. 1970) (greeting cards).

[100] *See, e.g.*, Sid & Marty Krofft Television Prod., Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977).

[101] *See, e.g.*, Atari v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607 (7th Cir.) *cert. denied*, 459 U.S. 880 (1982); Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125 (D.N.J. 1982). The videogame copyright cases, unlike the software copyright cases, lend themselves to idea/expression separation by visual comparison. *See generally* Nimmer & Krauthaus, *supra* note 36, at 39–41.

[102] *See* Knowles & Palmieri, *supra* note 81; at 128; Nimmer & Krauthaus, *supra* note 36, at 31; *see also* Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971) (classification into ideas and expression, at least in close cases, may simply state the court's desired result). *But see* Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1163 n.6 (9th Cir. 1977) (noting that the idea-expression dichotomy has been criticized as being results-oriented, affirming faith in the doctrine itself, and stating that the criticism may be alleviated by courts "being more deliberate in their consideration of this issue").

See *infra* Part IV for a discussion of the idea-expression separation as it applies to programs.

[103] Knowles and Palmieri, *supra* note 81, at 126 (emphasis omitted).

[104] Nimmer & Krauthaus, *supra* note 36, at 31.

[105] *Id.*; *see* Chafee, *supra* note 56, at 506–14.

[106] Nimmer & Krauthaus, *supra* note 36, at 32.

[107] *Id.*

[108] *See, e.g.*, Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1229–31 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987); *see also* R. Saltman, *supra* note 2, at A-10–A-11, A-62–A-65 (noting a need to provide technical expertise to the judiciary).

[109] Proof of access plus substantial similarity raises an inference that the defendant used the plaintiff's work. See *supra* Part II C.

ideas from the plaintiff's copyrighted work.[110] To evaluate these contentions, judges and lay factfinders must have some understanding of what programs do, how they are developed, and how they can be copied.

A. *The Nature of Computer Programs*

The Copyright Act 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."[111] Computers are very quick and accurate at executing steps that are time-consuming and error-prone for humans.[112] A computer's circuitry, however, called a processor or central processing unit (CPU), must be given very precise instructions as to what to do,[113] in what order, and with what data.[114]

The instructions to a computer, or program, must be given to the computer in the form of "machine language" notation.[115] Machine language is, however, difficult for

---

[110] See *supra* Part II D for a discussion of the idea/expression dichotomy.

[111] 17 U.S.C. § 101 (1982). This definition is fairly close to standard technical definitions of the term "program". Note, *supra* note 4, at 500 n.10.

[112] J. SOWA, CONCEPTUAL STRUCTURES: INFORMATION PROCESSING IN MIND AND MACHINE 27 (1984). Conversely, computers are not easily instructed to recognize patterns that are easily recognized by humans — identification of faces, for example. *Id.*

[113] The processor or CPU is the "brain" of the computer. The CPU is usually located on one microcircuit chip. Low-priced home computers were made possible by the development of microprocessors. Prior to the advent of personal computers in the late 1970s, most computer users were technically-trained people. REPORT, *supra* note 6, at 48.

As technology progresses, the line between hardware and software is disappearing. Today, microcode often serves as a substitute for elements of a computer's hardware circuitry. Samuelson, *supra* note 27, at 677. "Microcode" (also known as "firmware") defines what specific actions a computer will take in response to a computer-readable instruction. Yoches, *supra* note 37, at 357. The use of microcode allows a machine manufacturer or subsequent purchaser to change a primitive function of the machine without disassembling and reconstructing the hardware. Samuelson, *supra* note 27, at 677. Microcode is often called "firmware" because, though it is a program, it is considered a more integral part of the machine than software. *Id.* "Firmware" refers to a program that has been put into the computer's microcircuit chips by the manufacturer. The chips holding "firmware" are called "Read Only Memory" (ROM). Memory chips which the computer's user can write on for storage during program operation are called "Random Access Memory" (RAM). Note, *supra* note 6, at 415.

For more on the intellectual property protection afforded microcode, see Harris, *Apple Computer, Inc. v. Franklin Computer Corp. — Does a ROM a Computer Program Make*, 24 JURIMETRICS 248 (1984) [hereinafter Harris, *ROM*]; Harris, *Legal Protection For Microcode and Beyond: A Discussion of the Applicability of the Semiconductor Chip Protection Act and the Copyright Laws to Microcode*, 6 COMPUTER L.J. 187 (1985) [hereinafter Harris, *Legal Protection for Microcode*]; and Steinberg, *NEC v. INTEL: The Battle Over Copyright Protection for Microcode*, 27 JURIMETRICS 173 (1987). A microcode copyright infringement case is pending (NEC Corp. v. Intel Corp., No. C-84-20799-WAI, N.D.Ca., San Jose Div.). The Semiconductor Chip Protection Act of 1984 (SCPA), 17 U.S.C. § 901-14 (Supp. II 1984), gives owners of "mask works" the exclusive right to reproduce the mask work and to import or distribute a semiconductor chip product in which the mask work is embodied. 17 U.S.C. § 905 (Supp. II 1984). The SCPA's protection more closely resembles copyright than patent. Harris, *Legal Protection for Microcode*, *supra*, at 202.

[114] "Data" are representations of information either known at the start of a program's execution or obtained during the execution. Yoches, *supra* note 37, at 356.

[115] A computer may be described as essentially a series of switches, each with two positions, on or off. The computer's numbering system is binary, or "base two", reflecting the fact that a switch can only be on or off. C. METCALF & M. SUGIYAMA, COMPUTER BEGINNER'S GUIDE TO MACHINE LANGUAGE ON THE IBM PC & PC. JR. 5 (1985).

humans to comprehend.[116] Generally, instead of writing machine language instructions that the processor can execute directly, programmers write programs in a programming language, which then is translated mechanically to machine language by a compiler program.[117] Compilers make it possible for a programmer to write at a level commensurate with the capabilities of human expression, in a "high-level" language that uses simple English-like statements.[118] High-level languages are usually easier for novice programmers to learn than are the "low-level" languages that require that the programmer's approach to a problem more closely resemble the computer's functioning.[119] These high-level languages have their own syntactic and semantic rules, just as English does.

Some programmers write in assembly languages rather than in high-level programming languages. Assembly language is a form of machine code that humans can read. A programmer who writes a program in assembly language must approach the problem-solving task in the same systematic fashion that the computer will use. A program written in assembly language is converted into machine code by an assembler program. Assembly language programs generally run faster than higher-level language programs.[120]

Programs written in either high-level programming language or assembly language are called "source code" programs.[121] The machine-language version of a program is referred to as an "object code" program.[122] Although programs are generally written in source code, mass-marketed programs are generally licensed[123] or sold only in object code form in order to protect trade secrets and make imitation and unauthorized

---

[116] Yoches, *supra* note 37, at 356. For examples of machine language programs, see C. METCALF & M. SUGIYAMA, *supra* note 115, at 311–61.

[117] A compiler is a complex program that converts a program expressed in more or less human logic by a program designer into machine logic. The conversion can also be done by an interpreter program as the program is being run. Grogan, *Decompilation and Disassembly: Undoing Software Protection*, 1985 COMPUTER LAW ANNUAL at 15–16.

[118] Two "high-level" languages in common use are BASIC and FORTRAN.

[119] Grogan, *supra* note 117, at 15. For more on high level languages, see Ledsinger, *Copyright Protection of Object Code Computer Programs: Can Courts Determine Copying*, 9 COMM/ENT L.J. 255, 258 (1987); Yoches, *supra* note 37, at 356.

[120] Grogan, *supra* note 117, at 17. A recent ad by Microsoft for its Macro Assembler Version 5.0 starts with the statement "[y]ou probably already know that assembly language subroutines are the smartest way to get the fastest programs." PC MAGAZINE, Oct. 13, 1987 (back cover).

[121] Ledsinger, *supra* note 119, at 258.

[122] Yoches, *supra* note 37, at 356. Compilers and interpreters translate high-level source code to object code, whereas assemblers translate assembly language programs to object code. *Id.* at 356–57.

[123] A license is a grant of use rights. Many programs are licensed rather than sold in order to avoid the loss of product control that results from the "first sale" doctrine of copyright law. *See* 17 U.S.C. § 109 (1982). Under the "first sale" doctrine, the owner of a copyrighted item (a book, a diskette) has the right to transfer ownership or possession of that item. T. SMEDINGHOFF, *supra* note 36, at 55. If a software distributor wants to impose restrictions on product use and transfer to protect trade secrets, the distributor will license the use of the software subject to termination on breach of a restriction by the licensee. *See id.* at 55–56, 76–77, 83, 85–86, 166–68. A sample license agreement appears in Smedinghoff at 362–68. For additional sample clauses, see AMERICAN PATENT LAW ASSOC. CONTINUING LEGAL EDUCATION INSTITUTE ON THE LAW OF COMPUTER-RELATED TECHNOLOGY, COMPENDIUM OF PACKAGED SOFTWARE LICENSING PROVISIONS (1984).

Distributors of mass-marketed software often put "shrink wrap licenses" on packages of the product. Such a "license" generally states that the distributor has imposed certain restrictions on the use of software, restrictions which the "licensee" (buyer) accepts by the act of opening the package. These licenses may be unenforceable. T. SMEDINGHOFF, *supra* note 36, at 168–69.

modifications more difficult.[124] Machine-readable object code, incomprehensible to people,[125] consists of a string of ones and zeros, which are the only two symbols a digital computer can understand. Although a disassembler can assist in deciphering a program from its object code, a disassembler only roughly translates the object code into assembly code, not into the higher level language in which the program most likely was written.[126]

Personal computers generally are sold with operating systems programs already in place. Operating systems programs are fundamental programs that manage the internal flow of data within the computer, coordinate the functioning of the hardware, and allow the user to use the task-specific programs known as applications programs.[127] Operating systems software often is placed on semi-conductor chips ("ROMs") and installed directly into the computer's circuitry.[128] The user adds applications programs to perform such specific tasks as word processing, accounting, or game-playing.[129] The operating systems program enables the computer to execute application programs and to access systems resources so an operating systems program must be used with an applications program.[130] Applications programs are written for use with a specific operating systems program.[131] Because there is no industry-wide standard on the configuration of operating systems programs, applications programs written for use with one operating systems package generally will not run on computers using other operating systems programs.[132] Another manufacturer's personal computers are "IBM PC-compatible," for example, if those

---

[124] Grogan, *supra* note 117, at 5; Note, *supra* note 4, at 501–02. Object code also runs more efficiently on the user's computer. It need not go through the intermediate step of compilation or assembly. Note, *supra* note 4, at 501.

[125] Grogan, *supra* note 117, at 7.

[126] *Id.* at 18–21.

[127] Grogan & Kump, *The Broader Meanings of* Apple v. Franklin *in the Development of Compatible Operating Systems and In Determining Standards for Injunctive Relief*, 1985 COMPUTER LAW ANNUAL at 106–07. Examples of operating systems programs are Data General's RDOS, IBM's PC-DOS, and Microsoft's MS-DOS.

Microcode, discussed *supra* note 113, like operating systems programs, controls the internal functioning of the computer and enables the user to run applications programs. Microcode, however, directs the primitive functions of the computer while the operating systems program directs the relationship between the hardware and the application programs. Samuelson, *supra* note 27, at 678. For example, IBM's "BIOS" (Basic Input/Output Systems), an assembly language program that is part of the operating systems software used in IBM PCs, is designed to "provide an operational interface to the system and relieve the programmer of the concern about the characteristics of hardware devices." Davis, *IBM PC Software & Hardware Compatibility*, 1985 COMPUTER LAW ANNUAL at 123 (quoting the IBM Technical Manual).

[128] Ledsinger, *supra* note 119, at 259. An operating systems program might handle tasks such as creating the video display, checking the keyboard to see which keys are being hit, and storing programs. Note, *supra* note 6, at 414.

[129] Grogan & Kump, *supra* note 127, at 107. Videogames are computer programs with intricate screen displays for user play or interaction.

[130] Note, *supra* note 4, at 502. In the early days of computers, computers were task-specific. A computer was "hard-wired" for a particular function. The only way to change the function was to change the wiring. The development of a computer that would store encoded instructions ("programs") applicable to particular tasks eliminated the necessity of making hardware changes and the necessity of having different machines for different jobs. The first commercial programmable computer was manufactured in 1951. Samuelson, *supra* note 27, at 673–74 & nn.33, 34, 35.

[131] Grogan & Kump, *supra* note 127, at 107.

[132] *Id.*

computers can run, without modifications, applications software designed to run on IBM PCs.[133]

B. *The Design of Programs*

No one should attempt to separate a computer program into ideas and expressions without understanding how computer programs are designed.[134] In order to analyze software copyright problems, courts should have an understanding of the process by which a programmer arrives at the actual programming language version of a program. Courts need not, of course, have the level of knowledge necessary to actually design and code a program.

The beginning point in software design is identifying the problem to be solved. Computers tackle such problems as "check word spellings for correctness," "assist in checkbook balancing," and "handle payroll."[135] The culmination of software design is the creation of a programming language set of instructions to the computer using algorithms, which are similar to recipes. Algorithms specify explicitly and without ambiguities the steps the computer is to carry out on data to reach solutions.[136] The

---

[133] Many applications programs already have been written for the popular IBM PC and Apple PCs. Manufacturers of other PCs generally want their computers to have the capability of using the applications software written for the IBM and Apple computers. The competitive machines' operating systems software must, if compatibility is to be achieved, mimic the functions provided by IBM's copyrighted operating systems software. Davis, *supra* note 127, at 123.

It is possible to subdivide programs into further classifications — custom-designed versus mass-marketed, or main-frame versus microcomputer. The focus of the creator's major development effort will vary depending on the program's ultimate purpose. In the large computer market, for example, the major effort will be the development of the mathematical process used in the program. By contrast, videogame programs use well-known algorithms and will succeed only if the end user finds the program easy to use and fun. Keplinger, *supra* note 4, at 486–87.

[134] *Cf.* Reback & Siegel, *Toward a Comprehensive Test for Software Copyright Infringement*, 1985 COMPUTER LAW ANNUAL at 139, 145–46 (stating that the widespread use of top-down design, discussed *infra*, for programs facilitates the use of the Hand-Chafee-Nimmer "levels of abstraction" analysis, discussed *supra* in section II D).

The word "design" is more appropriate here than the word "written", for reasons discussed *infra* in this section. Briefly, the actual writing of program code in high-level or assembly language is only a small part of the process of creating a program to solve a particular problem. Computer industry writers use the word "design". *E.g.*, S. ALAGIC & M. ARBIB, *supra* note 1; B. LIFFICK, THE SOFTWARE DEVELOPER'S SOURCEBOOK (1985); I. SOMMERVILLE, SOFTWARE ENGINEERING (1985). Software "engineering" is concerned with building software systems that are larger than would be tackled by an individual and involves the use of engineering principles. SOMMERVILLE, *supra*, at 1–2. The term "software engineering" was introduced in the late 1960s at a conference on the "software crisis." The "crisis" was the realization that "third-generation" computer hardware made "heretofore unrealizable applications a feasible proposition." *Id.* at 1.

[135] The "problem identification" will, most likely, be stated only very broadly by the client, the future user of the program, thus requiring extensive detail-level problem identification before a program or programs are designed. Prior to actual program design, a systems analyst may determine how the existing system works and prescribe a computerized replacement. B. LIFFICK, *supra* note 134, at 2–6.

[136] S. ALAGIC & M. ARBIB, *supra* note 1, at 1. A more technical definition provides that an algorithm consists of a number of actions to be performed on data in a specified order, with possible repetitions, under stated conditions. *Id.*

Program coding is not really the end point of the design process. Documentation (user's manuals

underlying algorithms for a "spell check" program, for example, instruct the computer to compare words in a user-prepared document against words in a dictionary residing in the program's memory. If a word matches a dictionary word, the program deems it correctly spelled.[137] If the document word does not match a dictionary word, the program displays the document word on the user's screen so that the user can judge the word's correctness in spelling.[138]

Experts agree that software design is a creative process that programmers learn more through practice than from books, a process that cannot be formulated as a set of rules.[139] The end product, the program, is generally the result of numerous conscious choices by the programmer.[140] Though in the past, programmers often designed software through an ad hoc process characterized by hasty informal pre-coding development[141] and by attempts to formulate algorithms directly into a programming language.[142] Experts now recommend a more structured approach to program design.[143] Formerly, programmers would analyze a problem, draw a flow chart of the steps involved in the problem's solution, and then go directly to writing code. Today, however, experts consider flowcharts to be inadequate vehicles for expressing the design of a system.[144] The current trend is for a programmer to fully formulate the program's algorithms without worrying about the constraints of the particular programming language in which the program will ultimately be coded.[145] Ideally, the programmer should develop the details of the program's logic flow before actually beginning to code.[146]

---

and instructional material for future programmers working with the program) must be prepared. Frequently, software designers prepare documentation after the program itself has been coded. *See* B. LIFFICK, *supra* note 134, at 9. Maintenance (debugging, modifications to fit user needs) is required after a program is made available to users. *See id.* at 9–10; I. SOMMERVILLE, *supra* note 134, at 2–3. The software must be tested, and, if what is being designed is a system of programs, the integrated system tested before the system is made available to the user. I. SOMMERVILLE, *supra* note 134, at 3.

[137] I. SOMMERVILLE, *supra* note 134, at 43. The "spell check" function will not pick up mistakes in semantics. If, for example, the user mistakenly uses the phrase "typing arrangement" instead of "tying arrangement", the semantically incorrect word "typing" *will* match a memory dictionary word. One writer has described spelling programs as "a vain attempt to impose the clarity and logic of machine language on the disorder and ambiguities of human language." Mendelson, *Clonkers and Coolitude: Spelling Checkers Get Better*, PC MAGAZINE, Oct. 13, 1987, at 349.

[138] I. SOMMERVILLE, *supra* note 134, at 72.

[139] *See, e.g., id.* at 79.

[140] Sommerville distinguishes between well-designed systems that solve the problem and badly-designed systems. A well-designed system is straightforward to implement and maintain, easily understood and reliable. In contrast, although a badly designed system may work, it is unreliable, expensive to maintain, and difficult to test. *Id.* at 67.

In designing the user interface of the program, the designer must consider, *inter alia*, the characteristics of humans, users of the program. *See* B. LIFFICK, *supra* note 134, at 188–223.

[141] I. SOMMERVILLE, *supra* note 134, at 67.

[142] S. ALAGIC & M. ARBIB, *supra* note 1, at 1.

[143] *See, e.g.*, S. ALAGIC & M. ARBIB, *supra* note 1; B. LIFFICK, *supra* note 134; I. SOMMERVILLE, *supra* note 134.

[144] I. SOMMERVILLE, *supra* note 134, at 67.

[145] S. ALAGIC & M. ARBIB, *supra* note 1, at 1; B. LIFFICK, *supra* note 134, at 64–67; I. SOMMERVILLE, *supra* note 134, at 75–76, 106.

[146] B. LIFFICK, *supra* note 134, at 66. There are several reasons for favoring language-independent program design: (1) programming languages possess characteristics that do not always lead to desirable programs; (2) the correctness of the solution can be judged better by demonstrating that the logic in a program is correct than by executing lines of code on various data; (3) the design of

A programmer begins designing a program to handle a particular task by studying the over-all problem to be tackled, just as an attorney has an initial conference with a new client to begin to explore the client's legal needs. The programmer then, according to the tenets of the most popular design methodology, "top-down design," breaks the over-all problem into subproblems.[147] Top-down design requires the program's designer to "decompose the over-all problem into precisely specified subproblems, and prove that if each subproblem is solved correctly, and these solutions are fitted together in a specified way, then the original problem will be solved correctly."[148] The "top" of the design is the general description of the over-all problem. Levels of detail are successively added below this top level. Each lower level of design is another step of refinement, and the process terminates when adding further detail would require the writing of programming language code.[149]

The top-down design principle should not be alien to lawyers and judges, for attorneys and judges in effect apply top-down design principles to legal problems. An attorney breaks a client's over-all legal problem into precisely specified subproblems, then breaks down each subproblem further until reaching a detailed solution to the client's problem. Suppose, for example, that the client is a start-up software developer seeking legal help for the first time. The over-all problem requiring the attorney's attention — providing adequate legal representation — is actually a number of subproblems which the attorney must, at the outset, identify. Examples of such subproblems include consideration of the optimal business form; review of any contracts already signed; advice on future contracts or negotiation-stage contracts; advice on matters concerning employees; advice on relationships with competitors; design of an intellectual property protection plan; and tax planning. The attorney, having subdivided the over-all problem, "provide adequate legal representation," into subproblems, must then decompose each subproblem into many smaller subproblems. The subproblem "design an intellectual property protection system," for example, requires the attorney to consider what copyright protection, trademark protection, patent protection, and trade secret protection might be available to the client. The attorney then must consider each of these possible avenues of intellectual property protection at a detailed implementation level, explain the options to the client, and implement them through such steps as filing registration forms, affixing notices to the product, and drafting confidentiality agreements.

---

a program usually requires many drafts, and it is easier to change and edit the design before the design is implemented at the level of detail required for coding; and (4) once the design has been completed, the design can be coded into one or several programming languages. S. ALAGIC & M. ARBIB, *supra* note 1, at 2; B. LIFFICK, *supra* note 134, at 65–66.

[147] I. SOMMERVILLE, *supra* note 134, at 69; Reback & Siegel, *supra* note 134, at 146. S. ALAGIC & M. ARBIB, *supra* note 1, is devoted in its entirety to the principles of top-down design, as are two earlier works, L. L. CONSTANTINE & E. YOURDON, STRUCTURED DESIGN (1979) and E. YOURDON, MANAGING THE STRUCTURED TECHNIQUES (1979). Sommerville describes two other design methodologies, data-driven design (used mostly in relatively small data processing projects) and object-oriented design. I. SOMMERVILLE, *supra* note 134, at 69.

[148] S. ALAGIC & M. ARBIB, *supra* note 1, at 2. Top-down design also is called "stepwise refinement", indicating that there will be many rounds of approaching the problem, breaking it down into further details (a step of refinement), and then doing that again. B. LIFFICK, *supra* note 134, at 31.

[149] *Id.* at 31.

Litigators and judges also use top-down design principles. Take, for example, the attorney seeking a preliminary injunction for a copyright owner against the distributor of an allegedly infringing work. To solve the over-all problem, "get a preliminary injunction," the attorney must first identify what he or she is required to show. The attorney must break down each requirement into further subproblems. One subproblem, for example, is determining what suffices in a copyright case to establish likelihood of success on the merits. Beyond that, once the attorney has some case law indicating what, in a copyright case, establishes likelihood of success on the merits, the attorney must consider the facts of the particular case and apply the facts to determine the likelihood of succeeding on the merits. The judge in the preliminary injunction case will engage in his or her own top-down problem-solving by asking himself or herself a series of increasingly focused questions: (1) What is the nature of this motion? (2) What is the nature of the underlying cause of action? (3) What must a plaintiff show to get a preliminary injunction? (4) As to each of the elements of a preliminary injunction, what generally is sufficient in a copyright case? and (5) What has the plaintiff presented and how did the defendant respond?[150]

Programmers following top-down design principles solve problems by using the fundamental human problem-solving facility, abstraction;[151] so do attorneys. Abstraction is the process of considering an idea as an abstract entity without considering the details of how that entity actually is realized.[152] The programmer designing a spell-check program will identify the fundamental operations needed, such as split the document into words, sort the words, and compare the words with the program dictionary's words. Then the programmer will refine each component operation into its fundamental operations. The attorney's end product may include letters to the client, copyright and trademark registration forms, letters to third parties, contracts, and memos to the attorney's client file. The programmer's end product will be a program and documentation. Just as the attorney will produce much paper between the intake stage and the resolution of the client's over-all problem, the program designer will produce a series of increasingly-detailed abstractions in going from the identification of the gross features of the solution to step-by-step instructions for the computer.[153]

What these programmers create are instructions for the computer to follow. The programs contain algorithms — recipes spelling out the steps to be carried out on problem data to reach a correct solution.[154] Selecting the algorithms used to solve subproblems is an important part of the program design process. People use algorithms to perform many of their daily functions. Imagine, for example, explaining to a guest how to make coffee in a home coffeemaker. The coffeemaking algorithm includes instructions about where to put the water, how much water to use, where to find the coffee, where to find the filters, where the filter goes in the coffeemaker, where the coffee goes, how much coffee is needed, and what buttons to push.[155] Computers, lacking human intuition,

---

[150] These questions are loosely drawn from Plains Cotton Coop. Assoc. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256 (5th Cir.), *cert. denied*, 108 S. Ct. 80 (1987).

[151] I. SOMMERVILLE, *supra* note 134, at 79.

[152] *Id.*

[153] *Id.* at 80.

[154] S. ALAGIC & M. ARBIB, *supra* note 1, at 1.

[155] A recipe for food preparation is another example of an algorithm. B. LIFFICK, *supra* note 134, at 82.

must be given very precise instructions as to what to do when and on what data to do it. Because a computer can only follow algorithms expressed in some programming language, the final level of abstraction in the design stage must be coding the chosen algorithms in the chosen programming language.[156] Although the actual coding requires care, the major intellectual effort involved in program design is in the development of sufficiently detailed algorithms, not in the coding.[157]

The actual code version of the program usually is divided into sections called modules. A module performs exactly one function,[158] such as splitting the user's document into words. The programmer generally defines the modules at the preliminary design stage,[159] just as an attorney might outline the issues to be addressed in a brief. Ideally, modules are independent of one another, so that a change or mistake in one module will not affect another module's functioning.[160] Any subfunction used repeatedly by the overall program will be placed in a subroutine, which is a section of code that performs a certain subfunction and handles that subfunction without further inquiry as to the details.

The actual code version of a program will reflect the peculiarities of the chosen programming language. Code looks peculiar to those who are unfamiliar with programming language. Suppose, for example, that one wants to instruct an Apple II computer to clear the screen and print the words "This is a Test." The instructions in the BASIC programming language to accomplish that goal look like this:

```
5 HOME
10 PRINT "THIS IS A TEST"[161]
```

The source code for this task also could be programmed in assembly language, which would require many more lines of code resembling ordinary English even less than does the BASIC code.[162] The machine code version of the program, which the computer's compiler, interpreter, or assembler produces from the source code, would require even more lines of instruction because a number of machine instructions are generated for each source code instruction.[163] The binary code that the computer actually uses consists of ones and zeros, representing the electronic impulses, "on" and "off," that the computer can read.[164]

---

[156] *See* S. ALAGIC & M. ARBIB, *supra* note 1, at 12.

[157] *Id.* Algorithms chosen should be efficient as well as functionally correct. *Id.* at 14.

[158] B. LIFFICK, *supra* note 134, at 29.

[159] *Id.*

[160] *Id.* at 30. Much has been written about the process of writing or coding an actual program from the finished design. Some of the advice is language-independent, applicable regardless of the actual programming language being used — for example, the following suggestions for making programs more easily readable by humans: (1) Make a program more readable by naming program objects — constants, variables, procedures, and functions — with names that bring to mind the real-word entities being represented; (2) make a program more readable through careful design of layout — use of blank lines, indentation, word highlighting, and consistent paragraphing; and (3) limit module size to the number of lines that will be fit on two pages of source code listing. B. LIFFICK, *supra* note 134, at 129, 137; I. SOMMERVILLE, *supra* note 134, at 112–18.

[161] T. HARRIS, THE LEGAL GUIDE TO COMPUTER SOFTWARE PROTECTION: A PRACTICAL HANDBOOK ON COPYRIGHTS, TRADEMARKS, PUBLISHING, AND TRADE SECRETS 51–52 (1985).

[162] See *id.* at 53 for the assembly language version.

[163] I. SOMMERVILLE, *supra* note 134, at 144.

[164] T. HARRIS, *supra* note 161, at 55. A hexidecimal version of the binary code is actually made first within the computer. *Id.* at 54.

C. *How Programs Are Copied*

Software is easily duplicated mechanically. Because most operating systems allow users to copy a program from one storage device to another in a matter of seconds, it is easy to take a disk containing a program and duplicate the program onto another disk.[165] Only a few software distributors take measures to "copy protect" their programs,[166] and the ease with which software may be copied contrasts sharply with the difficulty and cost of developing software.[167] The software duplicators may have several different motives in duplicating the literal code of a program. The duplicator may desire to "get something for nothing" for personal use, desire to create copies to sell, or desire to create a computer that is fully compatible with some competing model.[168]

The more sophisticated software bootlegger may try to make his or her program look different from the copied program by changing variable names or changing the order of modules or subroutines. Also, a bootlegger can use a compiler or assembler to mechanically translate source code into object code.[169] Although mass-marketed programs rarely are distributed in source code form, a pirate could duplicate, compile, and sell in object code form a program originally distributed in source code. The pirate's program would not look like the copyrighted program, but it would still be a "copy" of the copyrighted program.

Present-day software copyright law is concerned not just with the outright duplicator, but also with the defendant who, without the copyright holder's authorization, created a clone of a copyright-protected program for a line of computers other than the computers for which the original program was designed. While the plaintiffs in the "first generation" software copyright cases alleged that the defendants mechanically duplicated a program's literal code, in the recent cases plaintiffs have claimed that the defendant copied the plaintiff's program's organization and structure to create a functionally-

---

[165] This duplication is much like the duplication of recordings with audio-taping equipment — a recording device does the work. Nimmer & Krauthaus, *supra* note 36, at 22. For more detail on mechanical duplication, see Samuelson, *supra* note 27, at 689–90. This mechanical duplication is a concern for software distributors on two levels: (1) small-scale copying for the use of friends who would otherwise have to buy their own copy; and (2) large-scale commercial copying, much of it done outside of the U.S. and brought here for sale in competition with the legitimate product. If a work has been duplicated in quantity in another country without the copyright owner's authorization, the copyright owner may be able to have the work excluded from this country under the Customs Act. *See* 19 U.S.C. § 1337 (1982). A copyright owner may also record the work with the U.S. Customs Service, which, upon spotting infringing works, will impound them. 19 C.F.R. §§ 133.21–133.24 (1987). *See* Blatt, *Battling Foreign High Technology Counterfeits Through U.S. Customs Enforcement*, 1985 COMPUTER LAW ANNUAL at 93.

[166] See Saltzberg, *Legal and Technical Protection Through Software Locks*, 5 COMPUTER L.J. 163 (1984) for a discussion of copy protection devices.

[167] *See* Yoches, *supra* note 37, at 357. Although hardware costs have fallen with the development of new technologies, software costs have not dropped. I SOMMERVILLE, *supra* note 134, at 1.

[168] Compatibility with a particular well-known computer model can be achieved only by using an operating systems program that mimics the functions provided by the well-known model's operating systems program. The defendants in two infringement suits brought by Apple contended that they had to duplicate Apple's operating systems programs to achieve compatibility. *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984); Apple Computer, Inc. v. Formula Int'l, Inc., 562 F. Supp. 775 (C.D. Cal. 1983), *aff'd*, 725 F.2d 521 (9th Cir. 1984). See *infra* Part IV for a discussion of *Franklin Computer*.

[169] *See* T. HARRIS, *supra* note 161, at 53–54 (noting that in such a case it would seem possible to show copying even if the original program was in BASIC and the copy in assembly language).

similar program in a different programming language. In the recent case *Whelan Associates v. Jaslow Dental Laboratory, Inc.*, for example, the plaintiff's program ran on IBM computers, which required application programs written in Event Driven Language (EDL).[170] The *Whelan* defendants created a program that would perform the same functions for the personal computers that use BASIC language application programs.[171]

The authors of two recent articles contend that, although legal restraints against unauthorized commercial reproduction and sale of software are appropriate, translations and other adaptations may benefit society by expanding the availability of technology, and that copyright law should not deter these efforts.[172] Generally, only the object code of the target program will be available, so the unauthorized translation or conversion must be done by "reverse engineering," a process involving the conversion of the original program's object code into a human-readable form that reveals the logic and structure of the original program.[173] The pirate can convert the object code into human-readable assembly language using a decompiler or disassembler.[174] The "reverse engineer" then must make many trial and error guesses about what was in the original high-level program, based on recognition of recurring patterns in the assembly language program.[175] Disassembling a complex program could take man-months or man-years.[176] A compiled program is likely to be even more difficult and time-consuming to reverse engineer than an assembled program.[177] Even if the original program's source code is available to the pirate, the pirate must expend effort to convert the program to a new programming language. Presently, translations from one high-level programming language to another cannot be done mechanically. There are additional difficulties in copying applications programs, because they originally are written for use with particular operating systems programs. A mechanically-translated applications program will not run on a computer using a different operating systems program from that used by the computer for which the original applications program was written. The pirate must not only create instructions in the new programming language, he or she must adapt the new program to the operating systems software, firmware, and hardware of the new computer.[178] Thus, preparing a translation of a program, whether the translator works from source code or object code, requires considerable effort on the part of the translator.

---

[170] 797 F.2d 1222, 1226 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

[171] *Id.*

[172] Karjala, *supra* note 36, at 54–59, 73–88; Nimmer & Krauthaus, *supra* note 36, at 36–62.

[173] "Reverse engineering," stated more generally, is the process of starting with a known product and working backward. Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974). *See* Grogan, *supra* note 117, at 12. Reverse engineering is common in many industries. *Id.* For a defense of reverse engineering of software, see Laurie & Everett, *Protection of Trade Secrets In Object Form Software. The Case for Reverse Engineering*, 1985 COMPUTER LAW ANNUAL, at 33–57.

[174] Grogan, *supra* note 117, at 17–21.

[175] *Id.* at 18–19. See *generally* Karjala, *supra* note 36, at 39–41 for a discussion of reverse engineering.

[176] Grogan, *supra* note 117, at 18–19.

[177] *Id.* at 19–21. It is unclear, at present, whether an infringing program produced through reverse engineering is a copy or a derivative work of the original program. A copy of the original program will, most likely, be made in the process of the reverse engineering. *Id.* at 25–26. This copying alone is an infringement of the copyright owner's exclusive rights. On the question of whether section 117 of the Copyright Act might authorize such copying, see *id.* at 28–30. *See generally* Conley & Bryan, *supra* note 36, at 599–602.

[178] *See* Samuelson, *supra* note 27, at 688. Nimmer and Krauthaus distinguish "value-added use"

IV. Distinguishing a Computer Program's Ideas From Its Expression

A. *The Two Principles*

In the 1970s Congress and the Commission on the New Technological Uses of Copyrighted Works (CONTU)[179] included computer programs in the category of works that contain expression that may be protected without granting a monopoly on underlying methods or ideas. A 1976 House of Representatives committee report explained the applicability to computer programs of Section 102(b)[180] of the 1976 Copyright Act, which codified the rule that copyright protection does not extend to ideas. The committee report stated that Section 102(b) would clarify that a computer programmer's expression is a copyrightable element of a computer program, but that the actual processes or methods embodied within the program are outside the scope of copyright law.[181] A few years later CONTU stated that the copyright statute should "make it explicit that computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright."[182] By 1983 the Copyright Office had issued a circular stating that "[c]opyright protection extends to the literary or textual expression contained in the computer program."[183]

The software copyright case in which the defendant has duplicated the plaintiff's program code in its entirety is relatively simple, in legal terms. Early software copyright cases were of this variety.[184] In such cases it is easy to conclude that the defendant took protected expression from the plaintiff's program. One who copies "the whole thing" copies the expression; literal similarity is always expressive similarity.[185] When the copy-

---

from piracy. Nimmer & Krauthaus, *supra* note 36, at 29. They define value-added use as use occurring when a second party uses the initial work as a basis but makes substantial modifications to create a new product. *Id.* at 30. Additionally, Karjala takes the position that courts should abandon the idea/expression distinction in software copyright cases and instead go directly to the policy question of whether what the defendant did was (a) reverse engineering for the purpose of improvement (lawful, in Karjala's view); or (b) socially undesirable piracy. Karjala, *supra* note 36, at 58.

[179] See *supra* note 33 for a discussion of CONTU's role in copyright legislation.

[180] Section 102(b) of Title 17 states as follows: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b) (1982).

[181] H.R. Rep. No. 1476, *supra* note 29, at 5670 (emphasis added). *See generally* 1 M. Nimmer, *supra* note 5, § 2.03[D].

[182] CONTU Report, *supra* note 33, at 2.

[183] Copyright Office's Circular R61 (May, 1983) (quoted in Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1242 n.38 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987)).

[184] *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984). In *Franklin* the defendant acknowledged that it had duplicated the plaintiff's software but maintained that the software was not protected by copyright. *See id.* at 1244. Later "outright duplication" cases include M. Kramer Mfg. Co. v. Andrews, 783 F.2d 421 (4th Cir. 1986); and Midway Mfg. Co. v. Strohon, 564 F. Supp. 741 (N.D. Ill. 1983).

[185] 3 M. Nimmer, *supra* note 5, § 13.03[A] at 13-35. *See supra* note 66 and accompanying text.

Even literal copying is not, however, necessarily infringement. First, if the idea and expression have merged, the copyist is free to take even the expression. *See supra* notes 82–89 and accompanying text. Furthermore, if the plaintiff's work was copied by the plaintiff rather than original to the plaintiff, the copyright is invalid. *See* Synercom Technology, Inc. v. University Computer Co., 462 F. Supp. 1003, 1009–10 (N.D. Tex. 1978). The originality required for copyright exists, however,

right defendant's program is not a verbatim copy of the plaintiff's work, determining whether the programs' similarities are idea-level or expression-level similarities is more difficult. Indeed, determining whether any similarities exist may be difficult; even source code programs may look like gibberish to judges, lawyers, and jurors.[186] Additionally, a programmer who works from a copyrighted program can easily make a new program that looks different from the copied work.[187] The copyist can make his program look different from the plaintiff's by compiling the plaintiff's source code program into object code,[188] by using a different compiler from that used by the plaintiff, or by making trivial changes in the original program.[189] The copyist could, for example, change variable names, reorder modules or subroutines, and add or delete comments.[190] Thus, the similarities between the plaintiff's program and "new" work may not be obvious to the factfinder. Furthermore, if the defendant has "translated" the plaintiff's program into a different programming language, the two programs will not resemble each other at all. Finally, if the defendant, in addition to translating the plaintiff's program into a new programming language, has borrowed only some of the features of the plaintiff's work and rejected others, the factfinder's job of determining whether there are not only similarities but also substantial expression-level similarities between the two programs becomes even more difficult.

The foreseeable difficulties involved in separating a program's unprotected ideas from protected expression, an issue that has emerged only in the 1980s, already has been the subject of extensive comment in the literature. A report of the Office of Technology Assessment (OTA)[191] concludes that the federal courts eventually will face a dilemma in software copyright cases. The report foresaw that the courts will either limit the protected expression in a program to literal code duplication, or copyright decisions inevitably will protect a "procedure, process, system, or method of operation" in violation of Section 102(b).[192] Another commentator, Karjala, questions whether courts ever will be able to efficiently separate ideas from expression in programs, because factfinders, who traditionally draw the line between ideas and expression, will need the expensive help of experts to draw this line for programs.[193] Karjala would eliminate the

---

if the author has introduced *any* element of novelty to previously created material. *Id.* at 1010. Nimmer and Krauthaus note that accepted programming styles and known subroutines could contribute to code similarity which, in their view, should not be actionable. Nimmer & Krauthaus, *supra* note 36, at 55.

[186] The question of whether similarities are similarities of idea or expression is a question of fact. SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 829 (M.D. Tenn. 1985). Often, however, judges have answered the question in copyright cases in a preliminary injunction context. Accordingly, the software copyright cases analyzed *infra* Part IV B are either bench trials or preliminary injunction orders.

[187] *OTA Report*, *supra* note 5, at 82; Conley & Bryan, *supra* note 36, at 582; and Note, *supra* note 4, at 513.

[188] Rarely, however, will the source code be available. *See supra* notes 123–24 and accompanying text.

[189] Conley & Bryan, *supra* note 36, at 582.

[190] Comments are statements that are not compiled, because they aid the programmer, rather than instruct the machine.

[191] *OTA Report*, *supra* note 5, at 81. For a rebuttal, see Baumgarten & Meyer, *Program Copyright and the Office of Technology Assessment*, 4 COMPUTER LAW. No. 11, 1 (1987).

[192] *OTA Report*, *supra* note 5, at 81.

[193] Karjala, *supra* note 36, at 55.

idea/expression analysis from software infringement cases, instead focusing the legal analysis on whether the defendant's use of the plaintiff's work conferred any substantial commercial advantage on the defendant.[194] Karjala reasons that if the defendant undertook reverse engineering at a cost approximating the actual development costs that the first developer did not recover during his or her original monopoly period, the defendant would not be able unfairly to undercut the original developer's price for the work.[195] Nimmer and Krauthaus contend if artistic expression is not at issue, that "only the most proximate copying and virtually complete similarity should be actionable."[196] The authors Conley and Bryan view the mere determination of similarities between two programs to be difficult, if not impossible[197] for lay factfinders, who will find difficult the determination of where the boundary between idea and expression lies.[198] Conley and Bryan would have the court focus on the defendant's conduct rather than the program alone to determine whether the defendant engaged in unauthorized copying.[199]

Presently, however, courts are attempting to apply the idea/expression dichotomy to programs[200] on a case-by-case basis, and some commentators support this approach. Davidson maintains that it is not really that difficult to distinguish between the ideas and expression of software.[201] Professor Goldstein assumes that courts can make the separation in computer program copyright infringement cases, stating that courts should hold the defendant liable for literally copying the surface details of the plaintiff's work, but should excuse the defendant who uses only functional elements of the plaintiff's work.[202] Several practitioners advocate using the traditional Hand-Nimmer abstractions approach[203] to software, determining, for each case, at what level of a program's abstractions copyright protection should begin.[204]

Courts could certainly avoid upcoming litigation difficulties by limiting copyright infringement for software to literal code duplication. This approach would be out-of-step, however, with pre-software copyright cases holding that two works need not be literally identical to be substantially similar.[205] This criticism is not dispositive; some commentators argue that software *should* be treated differently from traditional literary

---

[194] *Id.* at 56.

[195] *Id.* The competitor may, however, get a "free ride" by using the creator's advertising and image-development.

[196] Nimmer and Krauthaus, *supra* note 36, at 43. The authors approve the use of the idea-expression distinction as applied to programs resembling fiction works, such as videogames. *Id.* at 41. Their position regarding organizational copying is that such copying should be infringement "only if the duplicated structure encompasses the detail and entirety of the organization of a *complex* work." *Id.* at 52 (emphasis in original).

[197] Conley & Bryan, *supra* note 36, at 582.

[198] *Id.* at 587.

[199] *Id.* at 612–13.

[200] See *infra* Part IV B.

[201] Davidson, *supra* note 36, at 1082 & n.112.

[202] Goldstein, *supra* note 36, at 1124.

[203] See *supra* notes 75–81 and accompanying text for a discussion of the "abstractions" approach.

[204] Baumgarten & Meyer, *supra* note 191; Ladd & Joseph, *supra* note 11; Reback & Hayes, *The Plains Truth: Program Structure, Input Formats, and Other Functional Works*, 4 COMPUTER L. No. 3, 1 (1987); Reback & Siegel, *supra* note 134.

[205] *See, e.g.*, Sid & Marty Krofft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157, 1167 (9th Cir. 1977); Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 55 (2d Cir. 1936); Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930). *See generally* 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-20 & n.4; *supra* notes 66–74 and accompanying text.

works. Professor Karjala, for example, argues that the broad concept of "expression" used in traditional literary works infringement cases is inappropriate for computer programs because programs are technology, and copyright does not protect technology, absent some special policy justification.[206] Accordingly, Karjala concludes that only program copying not involving the time, effort, skill or expense of reverse engineering should be actionable.[207] Similarly, Nimmer and Krauthaus assert that "[c]onfusion results when copyright standards are applied to value-added use of technology."[208] Furthermore, copyright law, the authors of the *OTA Report* write, cannot successfully be applied to computer programs,[209] because it is impossible to distinguish clearly between idea and expression in a computer program by using traditional copyright analysis.[210]

It seems clear, though, that immaterial variations, such as changes easily made with a text editor, should not get the defendant off the hook in a software infringement case anymore than trivial changes get the defendant off the hook in a traditional literary work infringement case.[211] In the literary work case, paraphrasing is tantamount to copying.[212] Just as trivial dialogue variations do not save a defendant who pirated a play, trivial code variations should not allow a software copyright defendant to avoid a finding of infringement. Moreover, if a plagiarist may violate the copyright on a play without duplicating dialogue,[213] it would appear that a "clever copier" may violate the copyright on a program without duplicating or even paraphrasing program code. In the literary context, expression is not confined to the words of the protected work. Over forty years ago Professor Chafee observed that expression lies somewhere between an author's ideas and the manner in which that author wrote those ideas down.[214] This principle, applied to computer programs, would hold that a program's protected expression should not be limited to its literal code.[215]

The commentators who doubt the appropriateness of applying to software the broad concept of expression that is used in literary works infringement cases[216] appear to agree that the "clever copier" who makes only trivial changes in the code of a protected program is an infringer.[217] They object, however, to the idea that copyright protection on a program should extend beyond protection against close paraphrasing of the code.[218]

---

[206] Karjala; *supra* note 36, at 48–50.

[207] *Id.* at 54–57.

[208] Nimmer & Krauthaus, *supra* note 36, at 13.

[209] *OTA Report*, *supra* note 5, at 81.

[210] *Id.* at 83.

[211] 3 M. NIMMER, *supra* note 5, § 13.03[A], at 13-20.2 & n.9.

[212] Davis v. E.I. DuPont de Nemours & Co., 240 F. Supp. 612, 621 (S.D.N.Y. 1965).

[213] Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 55 (2d Cir. 1936). See *supra* Part II D.

[214] Chafee, *supra* note 56, at 513. Chafee was concerned with the application of copyright to motion pictures, the new technology of *his* time.

[215] Nimmer and Krauthaus contend that there is a serious risk that extending copyright protection beyond a program's code may over-protect the first author, imposing substantial restraint on future design and development. Nimmer & Krauthaus, *supra* note 36, at 48. The authors take the position that the original programmer "arguably" should control some adaptive works based on his work. *Id.* at 49.

[216] *See supra* notes 206–10 and accompanying text.

[217] *See* Karjala, *supra* note 36, at 40; Nimmer & Krauthaus, *supra* note 36, at 49.

[218] *See OTA Report*, *supra* note 5, at 81–85; Karjala, *supra* note 36, at 65, 75, 79; Nimmer & Krauthaus, *supra* note 36, at 52, 56, 61.

One argument against extending copyright protection beyond code duplication or close paraphrasing is that there is nothing in a program like a novel's expression; a program is not written with aesthetics or communication to a human in mind, it is written to instruct a computer.[219] Copyright protection covers not just literary or "arty" works, however; it covers that which is deemed expression, whether in fact works, contest rules, news reports, etc.[220] It is not surprising that it is impossible to define precisely "expression" for programs;[221] copyright law does not attempt to define the "expression" in any category of protected works. Instead, the line between idea and expression is drawn on an *ad hoc* basis.[222] The *OTA Report* statement that traditional copyright analysis is unhelpful in distinguishing between idea and expression in a computer program[223] is incorrect.[224] Courts may apply to programs the same "level of abstraction" analysis that they have, for years, used to draw the line between protected expression and unprotected ideas in traditional literary works.

The program design cycle itself[225] provides a strong basis for the position that protected expression should not be limited to actual code. The actual code in which a program is written is only the last step in the process of creating the work,[226] just as the words in which a novel is written are the last step in the novel-writing process. If the novelist's protected expression starts at a level more abstract than the words in his novel, the program designer's protected expression should start at a level more abstract than the final coding. Code and protected expression, then, are not synonymous. Accordingly, a program's expression lies somewhere between the program designer's ideas for solving the overall program problem and the code itself.

If courts extend copyright protection beyond a program's actual code, then courts will have difficulty determining whether the defendant borrowed protected expression or merely ideas from the plaintiff's work. A later programmer is free to use the ideas and methods used in a copyright-protected program. He or she need not work in ignorance of what has already been done.[227] If courts extend copyright protection beyond the literal code, there is a risk that they will protect ideas or methods, and thus overprotect early works. To avoid this overprotection, it is necessary to recognize that all programs contain many unprotected ideas.

---

[219] Nimmer & Krauthaus, *supra* note 36, at 41.

[220] *See supra* notes 93–101 and accompanying text.

[221] *See OTA Report*, *supra* note 5, at 81.

[222] *See supra* notes 76–81 and accompanying text.

[223] *OTA Report*, *supra* note 5, at 83.

[224] Baumgarten & Meyer, *supra* note 191, at 3.

[225] *See supra* Part III B.

[226] *See supra* notes 139–57 and accompanying text.

Where, one might ask, is the expression in the object code version of a program? Davidson, raising the question of "what is the legal relation between source and object code," concludes that source code and object code contain the same expression. Davidson, *supra* note 38, at 700–01.

[227] Nimmer & Krauthaus, *supra* note 36, at 38. One author has contended that progress in programming *requires* plagiarization, since progress comes in stepping-stone steps. Comment, *supra* note 50, at 1292. The *Whelan* court rejected that contention. Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1238 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

Reback and Siegel apply the principle of the legality of taking a program's ideas to the problem of creating an operating system program to achieve IBM PC compatibility. *See* Reback & Siegel, *supra* note 134. Additionally, Davis offers suggestions for creating a non-infringing operating system that will run the application programs designed for the IBM PC. Davis, *supra* note 127.

The observation that any program puts to work many unprotected ideas stems from understanding how people design programs.[228] Programs are designed by stating the overall problem and then identifying the underlying subproblems. The top-down design principle requires a programmer to divide the complex problem into smaller, less complex problems, and then to find a solution for each smaller problem. Next, the programmer must decompose each smaller problem down to its most specific, detailed form, the actual code.[229] The developer solves each smaller problem by putting an idea to work; thus, a program must use many ideas to solve the overall program problem.[230] The program's copyright protects none of the program's internal ideas.

Separating a program's internal ideas from its protected expression is difficult. The separation process requires the factfinder to look inside both the plaintiff's program and the defendant's program to determine and characterize what was taken by the defendant. Where the defendant's program resembles the plaintiff's, it is tempting to equate the unprotected ideas with overall program function, avoiding the necessity of looking inside. The Third Circuit Court of Appeals made this mistake in *Whelan Associates v. Jaslow Laboratory, Inc.*;[231] the decision has been criticized widely.[232] Limiting the unprotected ideas to the program's overall function, the highest "level of abstraction,"[233] overprotects the copyright owner by giving copyright protection to the internal program ideas. Copyright law, however, should reward the author of a work without giving the author a monopoly on the ideas used in the work. If courts limit the unprotected ideas to program function, everything in the program — all lower levels of abstraction — is then protected expression, unavailable for use by subsequent developers of programs. The *Whelan* court's error was not its decision to give copyright protection at a level of abstraction higher than literal code. Instead, the *Whelan* court erred by failing to correctly apply the necessary limiting principle, that ideas used in the program are not protected. If copyright protection for software extends beyond the literal code of a program, factfinders must look inside the program for the unprotected ideas that accomplish the overall program function. If the protected expression in a program extends beyond literal code, unprotected ideas must be sought in the program at a level less abstract than overall program function.

---

[228] *See supra* Part III B.

[229] *See supra* Part III B.

[230] A program designer starts with an abstract goal — "I want to design a program that will check the spelling of words" — and then breaks the abstract goal into more concrete concepts identifying subproblems. The programmer's mental processes concerning the desired spell-check program might be something like this:

> How do I use a computer to check the spelling of words in a user-created document? I do that by getting the computer to break the document into words, then by getting the computer to compare each document word with a memory dictionary word. I need to tell the computer to display on the screen any document word that does not match a dictionary word.

Once the programmer has identified the subproblems — divide document, compare document word with dictionary word, display non-matches — he or she will work with each subproblem until he or she figures out exactly how to get the computer to solve that subproblem, and, ultimately, the over all abstract goal of the program.

[231] 797 F.2d 1222, 1236 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

[232] *See, e.g.*, Karjala, *supra* note 36; Ladd & Joseph, *supra* note 11; Reback and Hayes, *supra* note 204.

[233] *See supra* notes 77–81 and accompanying text.

In Section IV B of this article, two principles — (1) don't confuse code with protected expression, and (2) the necessary limitation on that principle, don't confuse overall program function with the unprotected idea — are analyzed in the context of the computer software idea/expression cases decided to date. The recent Third Circuit decision, *Whelan Associates v. Jaslow Dental Laboratory, Inc.*, ignored the second principle, confusing the program function with the unprotected idea.[234] The *Whelan* court extended copyright protection beyond literal code but then, instead of looking inside the plaintiff's program for unprotected ideas, the court equated the unprotected idea with the overall program function.[235] The impact of the *Whelan* error is discussed in Section V of this article.

B. *The Software Copyright Infringement Cases: Separating Idea from Expression*

The idea/expression dichotomy first was raised in the computer program context in the 1978 case *Synercom Technology v. University Computing Co.*[236] In *Synercom*, both the plaintiff and the defendants marketed programs designed to help engineers perform structural analysis calculations.[237] The plaintiff did not allege that the defendants copied the actual code of the plaintiff's program. Instead, the plaintiff alleged that the defendants copied from the plaintiff's input formats the data input formats shown in defendants' user's manual, in violation of the plaintiff's copyright rights. The plaintiff provided its program users with input format cards, which guided the user in the arrangement and organization of data. The defendants' user's manual contained mirror images of the plaintiff's data input formats.[238] The commercial success of the plaintiff's program was largely attributed to the plaintiff's user's manuals and input formats, which made the plaintiff's program easier to use than other structural analysis programs on the market.[239] The defendants designed their structural analysis program to be "wholly compatible" with the plaintiff's program, meaning that the defendants' program received its data in the same manner as the plaintiff's program.[240] This "user interface" compat-

---

[234] *See Whelan*, 797 F.2d at 1236.

[235] *Id.*

[236] 462 F. Supp. 1003 (N.D. Tex. 1978) (bench trial order). The idea/expression dichotomy also has been addressed in numerous videogame copyright cases. In most of these cases the plaintiff alleges a copyright in the audiovisual component of the game. In such cases the court must determine whether such game features as character appearance and game play action are idea or expression. *See, e.g.*, Frybarger v. International Business Machs. Corp., 812 F.2d 525, (9th Cir. 1987); Midway Mfg. Co. v. Artic Int'l, Inc., 704 F.2d 1009, 1012, 1014 (7th Cir. 1983) (holding (1) that videogames are copyrightable as audiovisual works; (2) that a "speeded-up" version of a copyrighted game is a derivative work of the original game); Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 856 (2d Cir. 1982) (player participation does not disqualify videogames from copyright); Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, (D.N.J. 1982); Atari v. Amusement World, Inc., 547 F. Supp. 222, (D. Md. 1981).

Some of the videogame cases have involved alleged infringements of both the computer program that runs the game and the audiovisual component. *See, e.g.*, Williams Elec., Inc. v. Artic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982); Midway Mfg. Co. v. Strohon, 564 F. Supp. 741 (N.D. Ill. 1983) (both holding that the ROM, object code contained on a silicon chip, is protected by copyright).

[237] 462 F. Supp. at 1004. An IBM program performing the same general functions was on the market even prior to the plaintiff's creation of its program. *Id.* at 1006.

[238] *Id.* at 1012. The defendants' program used a preprocessor program that accepted data in these formats. *Id.*

[239] *Id.* at 1006.

[240] *Id.* at 1008–09.

ibility allowed former users of the plaintiff's program to switch to the defendant's program with a minimum of training and without losing data already accumulated on key punch cards.[241]

The *Synercom* defendants contended that the plaintiff's input formats were uncopyrightable blank forms.[242] Because copyright protects only an author's expression,[243] the defendants argued that material that contains no expression may not be the subject of copyright. Accordingly, blank forms, which contain no expression, are not copyrightable.[244] The court ruled, however, that forms that communicate information *may* be the subject of copyright protection.[245] Judge Higginbotham in *Synercom* concluded that the plaintiff's input formats "expressed," to the user, by placement of line, shading, and words, "what data to place where and how to do it" and therefore, that they were not blank forms.[246]

The *Synercom* defendants also argued that they had taken only the ideas expressed in the plaintiff's input formats.[247] Judge Higginbotham, considering that contention, avoided the error of identifying the unprotected idea with the function served by the input formats. Instead, Judge Higginbotham held that the defendants, in copying the ordering and sequencing of data from plaintiff's input formats, copied only unprotected ideas. This conclusion implicitly recognizes that the plaintiff's input formats contained unprotected ideas. Thus, the *Synercom* plaintiff's competitors were not limited to duplicating the function performed by the plaintiff's work; they could use the ideas that the plaintiff previously had used. The *Synercom* plaintiff argued that the ordering and sequencing of data were expression. Judge Higginbotham responded by asking what "separable idea" is expressed by the ordering and sequencing of data.[248] Had the judge viewed the unprotected idea as the overall function of the formats, all levels of detail beyond function would have been expression.[249]

---

[241] *Id.* at 1009. *Synercom* is, no doubt, the first of many "user interface compatibility" cases. User interface cloning may be the wave of the future as software distributors focus on designing software that is easy to use. The software of tomorrow is likely to arrive with on-line documentation and training rather than voluminous training manuals. Kellner, *Next On-Line: Users Are Calling For a New Face on Training*, PC WEEK, June 16, 1987, at 45.

[242] 462 F. Supp. at 1011.

[243] *See supra* Part II D.

[244] Aldrich v. Remington Rand, Inc., 52 F. Supp. 732, 736 (N.D. Tex. 1942); *Synercom*, 462 F. Supp. at 1011.

The principle that blank forms are not copyrightable usually is traced back to Baker v. Selden, 101 U.S. 99 (1879), discussed *supra* notes 59–63 and accompanying text.

[245] *Synercom*, 462 F. Supp. at 1011 (citing Harcourt, Brace & World, Inc. v. Graphic Controls Corp., 329 F. Supp. 517 (S.D.N.Y. 1971)). *See generally* 1 M. NIMMER, *supra* note 5, § 2.18.

[246] *Synercom*, 462 F. Supp. at 1012.

[247] *Id.*

[248] *Id.* at 1013. If Judge Higginbotham had viewed the ordering and sequencing of data as protected expression, the defendants had another fall-back argument, merger — because use of the idea of input formats requires substantial duplication of Synercom's arrangements, copyright protection should not be allowed. *Id.* at 1013. There were hundreds of programs available for structural analysis, all with data input formats. Only the plaintiff's and defendants' program used the input formats developed by the plaintiff. *Id.* at 1007. Obviously there were other ways to instruct users as to data input. The key to the merger defense is the level of abstraction at which the idea is defined — if the unprotected idea is simply the input format function, the merger argument must fail. If, however, the unprotected idea is *this particular way* of entering data, the argument would succeed.

[249] As an alternative basis for the conclusion that defendant's use of the plaintiff's input formats

In four later software idea/expression cases — *SAS Institute, Inc. v. S & H Computer Systems, Inc.*;[250] *Q-Co Industries, Inc. v. Hoffman*;[251] *Plains Cotton Cooperative Association v. Goodpasture Computer Service*;[252] and *Digital Communications Associates, Inc. v. Softklone Distributing Corp.*,[253] — the courts recognized, like *Synercom*, that a computer program contains many uncopyrightable ideas. *SAS*, *Q-Co*, and *Plains* involved alleged copying of the plaintiff's program code or program structure; *DCA*, like *Synercom*, involved user interface copying.

In *SAS* the plaintiff maintained that the defendant used the plaintiff's object code and source code to create a statistical analysis program similar to the plaintiff's.[254] The plaintiff's program ran only on IBM-compatible computers. The defendant "translated" or converted the program so that it would run on non-IBM computers. The plaintiff claimed that the defendant's product was either a "copy" of its program or a "derivative work" based on that program. In developing the new program, the defendant made printouts of the plaintiff's source code available for the defendant's programmers to examine.[255] The *SAS* court found that the defendant used the plaintiff's source code "extensively and systematically" in preparing the defendant's product. Also, although the defendant's program was not a duplicate of the plaintiff's, the court ruled that the translated program drew heavily from the plaintiff's program.[256] Further, the defendant prepared no design documentation for its software, which the court also considered evidence of wholesale copying.[257]

If the *SAS* court considered whether the defendant took unprotected ideas or protected expression, it is not apparent from the court's opinion.[258] If, as the judge

---

was not infringement, the court stated that "if the court is wrong in its finding that order and sequence are expressed ideas, not expressions, its alternative holding is that the formats are not copyrightable." *Id.* at 1014. Is the court, in stating that "the formats are not copyrightable," analogizing to the "blank form" cases? That seems unlikely, since, earlier in the opinion, the court rejected the defendant's contention that the formats were uncopyrightable blank forms. *Id.* at 1011–12. Is the court here stating that in an input format the idea and the expression have merged? If the court is stating that the ideas in the input formats are capable of only one expression, the statement is less than clear.

[250] 605 F. Supp. 816 (M.D. Tenn. 1985).

[251] 625 F. Supp. 608 (S.D.N.Y. 1985).

[252] 807 F.2d 1256 (5th Cir.), *cert. denied*, 108 S. Ct. 80 (1987).

[253] 659 F. Supp. 449 (N.D. Ga. 1987).

[254] SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 817 (M.D. Tenn. 1985). *SAS* was tried without a jury in the fall of 1983. *Id.*

[255] *Id.* at 821 (Finding 30). The programmers had text editors available to help them edit and rearrange the plaintiff's code. *Id.* at 822 (Finding 31). The text editor was used to systematically change code. *Id.* at 823 (Finding 44).

[256] *Id.* at 822. One of the plaintiff's experts testified that he had identified forty-four specific instances in which the defendant's program showed evidence of direct copying from the plaintiff's work. The defendant argued that a showing of forty-four examples of copying in the context of 186,000 lines of source code established, if anything, trivial similarities between the programs. The court rejected the defendant's contention. 605 F. Supp. at 822 (Findings 35 and 38). The defendant had destroyed earlier versions of its source code that were more similar to the plaintiff's code. *Id.* at 822–23 (Finding 40).

[257] *Id.* at 823–24 (Finding 46). In further support of its conclusion of copying, the court noted that the defendant's program contained functionless references that were present in the plaintiff's code. *Id.* at 824 (Finding 52).

[258] *See id.* at 825 (Finding 58) (stating that the copying was of expression, not merely ideas); *id.* at 829 (whether the similarities are of idea or expression is a question of fact, and the court found as a matter of fact that the defendant duplicated expression).

decided, the defendant extensively used the plaintiff's source code, the defendant probably should be viewed as having taken expression not just ideas. Even though the court did not identify the program's unprotected ideas or explain why what was taken qualified as expression, the court nonetheless avoided the error of equating the unprotected idea with program function. The *SAS* court stated that the defendant presented no evidence that the "functional abilities, ideas, methods, and processes of [the plaintiff's program] could be expressed in only very limited ways."[259] This statement indicates implicitly that the plaintiff's program contained many ideas. Here the court appears to have recognized that there are many unprotected ideas within the protected program.

The *SAS* court clearly recognized this article's principle that code and protected expression are not synonymous. The *SAS* defendant maintained that its statistical analysis program could not be substantially similar to the plaintiff's because the plaintiff had shown that the defendant directly copied from the plaintiff's program only forty-four lines of the defendant's 186,000 total lines of source code.[260] The court rejected that contention, finding instead that the defendant's copying "pervaded" the defendant's product.[261] The defendant, in the court's view, did more than copy specific lines of code; rather, the defendant also copied the "organization and structural details" of the plaintiff's work.[262]

Accordingly, the *SAS* court acknowledged that a program's expression is not limited to its code; organization and structural details also are protected expression. The *SAS* court made that point more explicit by citing *Meredith Corp. v. Harper & Row, Publishers, Inc.*[263] In *Meredith*, the court found that the defendant duplicated a textbook's expression by outlining the textbook and giving sections of the outline to writers who then wrote new text based on their assigned outline segments.[264] The *Meredith* court found infringement based primarily on the defendant's duplication of the plaintiff's topic selection and arrangement.[265] The *SAS* court found that the *SAS* defendant followed a procedure similar to that used by the *Meredith* defendant, picking a leading work and duplicating the basic outline of that work. The *SAS* defendant even provided portions of the plaintiff's source code to the programmers who wrote the defendant's code, the court noted.[266]

---

[259] *Id.* at 825 (Finding 60). For criticism of this court's emphasis on the defendant's bad faith, see Ledsinger, *supra* note 119, at 268–75; Radcliffe, *Recent Developments in Copyright Law Related to Computer Software*, 1985 COMPUTER L. REP. 189, 194. Another commentator views the case as suggesting that the courts may be moving toward the view that the defendant's course of development of its software is relevant to proving substantial similarity. See Note, *supra* note 4, at 515–16.

[260] 605 F. Supp. at 822, 830 (Finding 38).

[261] *Id.* at 830.

[262] *Id.* at 822, 830 (Finding 38). The court also noted that two of the plaintiff's employees, on observing earlier versions of the defendant's source code, found numerous instances of "literal, near literal, and organizational copying." *Id.* at 822 (Finding 39). The court concluded that the defendant had destroyed earlier versions of its source code to disguise its conduct. *Id.* at 824.

[263] 378 F. Supp. 686 (S.D.N.Y.), *aff'd*, 500 F.2d 1221 (2d Cir. 1974), (*slip op. after trial*), 413 F. Supp. 385 (S.D.N.Y. 1975).

[264] *Meredith*, 413 F. Supp. at 386.

[265] *See id.* at 386, 387. Only eleven percent of *Meredith* defendant's actual prose was copied from the plaintiff's work. *Id.* at 386.

[266] *SAS*, 605 F. Supp. at 830. The *SAS* case, as one commentator has suggested, demonstrates the difficulties of evidence and proof in software copyright litigation. By the time of trial the

The *SAS* court noted that substantial similarity of protected expression does not require literal identity — "a play may be pirated without using the dialogue."[267] The court adopted as a finding of fact the testimony of an expert concerning the nature of programming:

> [B]eginning with a broad and general statement of the overall purpose of the program, the author must decide how to break the assigned task into smaller tasks, each of which must in turn be broken down into successively smaller and more detailed tasks. At the lowest levels the detailed tasks are then programmed in source code. *At every level, the process is characterized by choice, often made arbitrarily, and only occasionally dictated by necessity. . . .* As the sophistication of the calculation increases, so does the opportunity for variation of expression.[268]

The court's conclusion that the defendants duplicated not just ideas, but also expression may have stemmed from a belief, based on this testimony, that choices made by a program author at detailed levels of the program preparation are expression.

In both *Q-Co*[269] and *Plains*,[270] the courts held the defendant did not infringe the plaintiff's copyright, although in each case the defendant's program was very similar to the plaintiff's program.[271] In each case the plaintiff alleged that its program was infringed by a program prepared by the same programmers who earlier had created the plaintiff's program.[272] The defendants in both cases obviously were completely familiar with the plaintiff's program. In neither case, however, did the defendant literally duplicate the plaintiff's work.

In *Q-Co* the court found that there were four modules common to the plaintiff's and defendant's programs, but ruled that the defendant had taken only ideas, not

---

defendant had destroyed early source code printouts. Expert witnesses testified based on reconstructed source code. The defendant apparently continued to modify its program during the course of litigation, eliminating previously spotted similarities. Radcliffe, *supra* note 259, at 194.

[267] *SAS*, 605 F. Supp. at 829 (quoting Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 55 (2d Cir.), *cert. denied*, 298 U.S. 669 (1936)).

[268] *Id.* at 825 (Finding 59) (emphasis added).

[269] Q-Co Indus., Inc. v. Hoffman, 625 F. Supp. 608 (S.D.N.Y. 1985).

[270] Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d 1256 (5th Cir.), *cert. denied*, 108 S. Ct. 80 (1987).

[271] The *Plains* court described the defendant's program as very similar to the plaintiff's "on the functional specification, programming, and documentation levels." 807 F.2d at 1259. The *Q-Co* defendant's program contained four modules corresponding to four of the twelve modules in the plaintiff's program. 625 F. Supp. at 614.

[272] *Q-Co*, 625 F. Supp. at 611; *Plains*, 807 F.2d at 1258. One of the individuals involved as defendants in *Q-Co* had formerly worked for the plaintiff and had developed a teleprompter conversion program for the plaintiff. After the defendant left the plaintiff's employment, the defendant and another programmer prepared a software program to turn an IBM PC into a teleprompter. The two programs had some differences arising out of the hardware differences between the IBM PC and Atari PC, the computer for which the original program was designed. The second program was written in a different computer language than the plaintiff's program and employed distinct algorithms. 625 F. Supp. at 611–15.

In *Plains* the individuals who created for the plaintiff a mainframe software system, designed to assist farmers in the growing and marketing of cotton, left the plaintiff's employment. They eventually went to work for the defendant, where they created a personal computer program serving the same purpose. 807 F.2d at 1257–59.

expression.[273] This ruling recognizes implicitly that the modules contained many unprotected ideas. The *Q-Co* court found the four corresponding modules to be "similar in structure and organization, including a few structural similarities."[274] The court stated that the plaintiff did not show that the defendant's program modules contained any unique expression derived from the plaintiff's work. Rather, the court found, the defendants reused only the ideas they had used earlier in designing the plaintiff's modules.[275]

In *Plains*, the Fifth Circuit upheld the district court's findings, made on a motion for a preliminary injunction, that the plaintiff did not demonstrate that the defendant had copied the plaintiff's protected expression.[276] The program designers testified that they did not, in preparing the defendant's personal computer program, refer back to the plaintiff's mainframe program, which they had earlier created; they merely reused their knowledge of the industry.[277] The Fifth Circuit refused to reverse the trial court, stating that the evidence indicated that many of the similarities between the two *Plains* programs were dictated by the externalities of the industry.[278] This statement may be an acknowledgement that many unprotected ideas go into the design of any program.

Whether *Q-Co* and *Plains* recognize that code and protected expression are not synonymous is questionable. The *Plains* appellate court, noting that the issue of whether the reproduction of a program's organizational structure was copyright infringement was presented only on a partially developed record from the denial of a preliminary injunction, "decline[d] to hold that [sequence and organization] patterns cannot constitute ideas in a computer context."[279] Similarly, in *Q-Co* there were no allegations of outright code copying. The plaintiff's and defendant's programs were written in different computer languages and used different algorithms.[280] The court found that the defendant's program was a different program *"because of language and hardware."*[281] In so stating, the court may have equated protected expression and programming code — the court's emphasis on the use of a different language may stem from a belief that protected expression is limited to code. The *Q-Co* court concluded that "the [plaintiff's and defendants'] modules in different languages were similar in the sense of ideas 'rather than expressions.'"[282] Although it is impossible to know, the court may have identified the program similarities as idea-level similarities simply because there was no code duplication.

---

[273] 625 F. Supp. at 616.

[274] *Id.* at 614.

[275] *Id.* at 616. The *Q-Co* court stated that "there is no testimony establishing any unique expression based on the existence of the . . . modules, since the same modules would be an inherent part of any prompting program." *Id.* at 616. Although the court seems to recognize the existence of unprotected ideas within the modules, the reasoning is puzzling. Is the court saying that the modules contain no expression because they contain no *novel* ideas? Such a conclusion would be incorrect. *See supra* notes 16–18 and accompanying text.

[276] Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d at 1256, 1262 (5th Cir.), *cert. denied*, 108 S. Ct. 80 (1987).

[277] *Id.* at 1260–61.

[278] *Id.* at 1262.

[279] *Id.*

[280] 625 F. Supp. at 614.

[281] *Id.* at 615 (emphasis added).

[282] *Id.*

The *Q-Co* court also addressed the plaintiff's claim that the defendants' program was a derivative work based on the plaintiff's program.[283] In this context, the court did consider "the relevance of similarities of [program] structure and organization,"[284] referring to *Synercom*[285] for guidance. Judge Higginbotham, questioning in *Synercom* whether the input formats copied by the defendants were protected expression, asked what separable idea was being expressed in the formats.[286] To answer that question Judge Higginbotham analogized to the "Figure-H" pattern of an automobile stickshift. Judge Higginbotham labeled the "Figure-H" pattern itself the idea. Expression of this idea, according to Judge Higginbotham, would take some form other than the stickshift itself: "The pattern . . . may be expressed in several different ways: by a prose description in a driver's manual, through a diagram, through a photograph, or driver training film, or otherwise."[287] Although each of these examples would be protected by copyright, Judge Higginbotham reasoned, the copyright on a form of expression would not prevent another car maker from using the Figure-H shift.[288] The *Synercom* analogy is disturbing in the context of software, however, because Judge Higginbotham seems to be equating protected expression with description of a functional object. If only descriptive or instructional material qualifies as expression, then programs do not contain expression.[289] Although prose descriptions of the software, user manuals, and reviews of software performance would qualify as protected expression, the functional object itself — the program — would be only an unprotected idea. The *Q-Co* court expanded the *Synercom* stickshift analogy, concluding that the program order and organization could "be more closely analogized to the concept of wheels for the car rather than the intricacies of a particular suspension system."[290] Thus it appears that the *Q-Co* court is holding that only complex programs qualify for copyright protection. The *Q-Co* court further noted that the *SAS*[291] court had considered the relevance of organizational similarities in the plaintiff's and defendant's programs, but it distinguished *SAS* because *SAS* involved "slavish copying" by the defendant.[292] The difference between the hardware used by the plaintiff and the defendant, the *Q-Co* court stated, would have made slavish copying impossible in this case.[293]

In *DCA*[294] the question before the court was whether the defendants' program infringed the plaintiff's audiovisual work copyright on the arrangement and design of

---

[283] *Id.* at 615–16.

[284] *Id.* at 616.

[285] Synercom Tech., Inc. v. University Computing Co., 462 F. Supp. 1003 (N.D. Tex. 1978). See *supra* notes 204–09 and accompanying text for a discussion of the facts of the case.

[286] *Id.* at 1013.

[287] *Id.*

[288] *Id.*

[289] Under this view the program comments — program statements that guide the programmer and are not translated into machine code — could be expression, but nothing else in the program would be.

[290] *Q-Co*, 625 F. Supp. at 616.

[291] 605 F. Supp. 816 (M.D. Tenn. 1985).

[292] *Q-Co*, 625 F. Supp. at 616.

[293] *Id.*

[294] Digital Communications Ass'n v. Softklone Distrib. Corp., 659 F. Supp. 449 (N.D. Ga. 1987). The original name of the case was Microstuff, Inc. v. Softklone Distributing Corp. In 1986 Digital Communications Associates purchased Microstuff. DCA was then substituted as the plaintiff in the action. *Id.* at 453.

a program's status screen.[295] The plaintiff's program, Crosstalk, enabled a user's computer to "communicate" with other computers, allowing a microcomputer user to access information stored in other microcomputers or in a remote mainframe computer.[296] Crosstalk was a commercial success.[297] The success was partly attributable to the program's "status screen," or main menu, which appeared immediately following the sign-on screen display to give the user the program's parameter/command terms under various descriptive headings and to allow the user to enter commands changing the program's operation.[298] The *DCA* defendant, ForeTec Development Corporation, obtained through commercial sources a copy of Crosstalk for purposes of developing a "clone" of Crosstalk.[299] ForeTec's legal counsel advised the company that though the Crosstalk program's source and object codes were protected by copyright, the status screen was not copyrightable. Accordingly, the attorney concluded that use of a similar or identical status screen should not constitute an infringement of the plaintiff's copyrights.[300] ForeTec then developed a Crosstalk clone called "Mirror," which it began marketing.[301]

After DCA filed suit and moved for a preliminary injunction, the defendants contended that status screens are not copyrightable because status screens are a necessary expression of the idea underlying status screens.[302] In order to analyze that contention, the court had to identify the unprotected ideas involved in the status screen. The court could simply have stated that the status screen's "idea" was that of communicating with the program's user. Since every program has a user interface, it would be easy, under that approach, to conclude that there are various means of achieving the desired purpose. Because there are various means of achieving the desired purpose of communicating with the user, the plaintiff's status screen would then be, in its entirety, protected expression.

The *DCA* court did not choose to identify the idea underlying the status screen as merely communicating with the user. Instead of defining the unprotected idea in terms of the overall function, the court defined the program's idea as "the process or manner

---

[295] *Id.* at 452. One of the defendants' defenses to the copyright infringement action was that the Crosstalk status screen was not copyrightable subject matter. As a preliminary matter, the court had to decide whether a program's copyright protection extends to the program's screen display: Does copying a program's screen display, absent copying of the source code, object code, sequence, organization or structure, infringe the program's copyright? Judge O'Kelley of the Northern District of Georgia determined in *DCA* that screen displays are not protected by the program's copyright. The court then went on to consider the plaintiff's claim that the defendants' copying of the plaintiff's status screen infringed the plaintiff's separate copyright on the status screen. *Id.* at 455–56.

[296] *Id.* at 452.

[297] *Id.*

[298] *Id.* The plaintiff obtained copyright registrations on the Crosstalk User Manual, the program itself and the "main menu" status screen. *Id.* at 454.

[299] *Id.* at 453.

[300] *Id.*

[301] *Id.* ForeTec formed a separate corporation, a wholly owned subsidiary, Softklone Distributing Corporation, for the purpose of marketing and distributing Mirror. *Id.*

[302] *Id.* at 457. The defendants, relying on Baker v. Selden, 101 U.S. 99 (1879), argued both that (1) the Crosstalk status screen is a necessary expression of its idea so as to merge expression with idea; and (2) that the status screen is nothing more than a "blank form" for recording the program user's choices of commands and parameters. The court rejected both contentions. 659 F. Supp. at 456–57.

by which the status screen . . . operates."[303] The expression, therefore, is "the method by which the idea is communicated to the user."[304]

In *DCA*, Judge O'Kelley identified ideas within the status screen. "Certain aspects" of the screen, he stated, "are clearly ideas which any other party, including the defendants, could legally copy."[305] The use of a screen to reflect program status is an idea, the court ruled, as is the use of a command-driven program and the typing of two symbols to activate a command.[306] The *DCA* defendants, however, took more than those ideas from DCA's program. Rather, the court stated, the defendant copied arrangement headings, capitalization, and highlighting from the plaintiff's screen, features that involved "considerable stylistic creativity and authorship above and beyond the ideas embodied in the status screen."[307] This copying was, in Judge O'Kelley's view, not necessary for use of the screen's ideas. Accordingly, the defendant's merger defense failed.[308] The court concluded that the defendant's screen was substantially similar to the plaintiff's at both the idea level and the expression level and granted the preliminary injunction.[309]

In *E. F. Johnson v. Uniden Corp.*, decided in 1985, a few months after *SAS* and before *DCA*, the defendant reverse engineered the plaintiff's software to create a trunked logic mobile radio compatible with the plaintiff's mobile radio system.[310] *Uniden*, like *SAS*, is a pre-*Whelan* "translation" case recognizing that code and protected expression are not synonymous. The *Uniden* defendant introduced into evidence a line-by-line, side-by-side comparison of the plaintiff's and defendant's codes translated into a common language. It was, to the court, "obvious at a glance that the programs thus depicted are not line-

---

[303] *Id.* at 458.

[304] *Id.* This definition of expression will not work for program code or program organization copying cases. In program code or organization cases, the program is designed to instruct the machine, so communication to the user cannot be the test for expression.

[305] *Id.* at 459.

[306] *Id.*

[307] *Id.* at 460.

[308] *Id.* The defendants relied on *Synercom*, arguing that the status screen, like the input formats in *Synercom*, was the necessary expression of the idea and thus merged with the idea. As Judge O'Kelley pointed out, however, the *Synercom* defendant did not create format cards with the same headings and shaded areas as the plaintiff's input formats. *Id.* As to the *Synercom* court's murky alternative holding that formats are not copyrightable, Judge O'Kelley in *DCA* found that though the *Synercom* format cards showed no stylistic creativity beyond data sequencing, the *DCA* plaintiff's status screen involved considerable stylistic creativity and authorship beyond the level of the ideas embodied in the status screen. *Id.*

As to the defendants' "blank forms" argument, the court concluded that the status screen, even if a "form," expressed and conveyed information and was therefore copyrightable. *Id.* at 462.

[309] *Id.* at 465. On August 10, 1987 the parties announced that they had settled the case and that the defendant would drop its appeal to the Eleventh Circuit. *Computer Industry Litigation Rep.*, August 24, 1987, at 6351–52.

Litigation in the "user interface" copying area continues, with two cases filed in 1987 by Lotus Development Corporation alleging that defendants copied the organization, structure and sequence of Lotus 1-2-3 and the "user interface" of Lotus 1-2-3, and with Apple Computer, Inc. v. Microsoft Corp., C.A. No. C 88-20149 (N.D. Cal., filed Mar. 17, 1988). The user interface litigation is referred to in the software industry as "look and feel" copyright litigation. See *supra* note 69 and accompanying text for the origin of that phrase. The *Lotus* cases are Lotus Dev. Corp. v. Mosaic Software, Inc., C.A. No. 87-0074-K (D. Mass. filed on Jan. 12, 1987); and Lotus Dev. Corp. v. Paperback Software Int'l, No. 87-0076-K (D. Mass., filed Jan. 12, 1987).

[310] E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1489–90, 1492 (D. Minn. 1985).

by-line duplications."[311] The court found the line-by-line code differences unconvincing, however, recognizing that the defendant's actual program code would have to look different from the plaintiff's code because the plaintiff and defendant used different microprocessors.[312] Literal translation of the plaintiff's program into a language compatible with the defendant's microprocessor, the court noted, "necessarily involves a skewing of the program such that line-by-line comparison becomes meaningless."[313] Additionally, the court recognized that one microprocessor will require a different number of commands than a second microprocessor.[314]

The *Uniden* court held that the defendant's program was substantially similar to the plaintiff's at the level of expression as well as at the level of ideas,[315] even though the defendant's code and plaintiff's code, when translated into a common language, did not look alike. Thus, the court recognized that expression begins somewhere between the general outline of the program and the coding. Additionally, the court stated in a footnote that the defendant's program would have been non-infringing if the defendant had "contented itself with surveying the general outline of the [plaintiff's program], thereafter converting the scheme into detailed code through its own imagination, creativity, and independent thought."[316] At some point in the program's design, the court noted, "the idea or 'broad and general statement of the overall purpose' of the program merges into the expression, the 'smaller and more detailed tasks' necessary to carry out that idea."[317] This statement implicitly acknowledges both that the plaintiff's program contained many unprotected ideas and that protected expression is not just the code itself. The court came close to applying a Hand-Nimmer abstractions test,[318] noting that a programmer designs a program by breaking the program's overall purpose into successively smaller tasks, and finally expressing each detailed task in source code.[319] The *Uniden* court did not go on to draw the line between idea and expression, however, because the defendant took virtually the plaintiff's entire program.[320]

---

[311] *Id.* at 1497.

[312] *Id.* The plaintiff's program was designed to run on an Intel microprocessor, whereas the defendant's program ran on a Hitachi microprocessor. *Id.*

[313] *Id.* The court noted that the district court in *Whelan*, despite finding that the defendant's BASIC program was not a line-by-line duplication of the plaintiff's EDL program, had concluded that copying had taken place. *Id.* (citing Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 609 F. Supp. 1307 (E.D. Pa. 1985), *aff'd*, 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987)). The *Uniden* opinion quoted *Whelan's* comment that "transferring or converting from one computer language to another is not comparable to translating a book written in English to French." *Id.* at 1497 (quoting *Whelan*, 609 F. Supp. at 1320).

[314] *Id.*

[315] *Id.* at 1501, 1503. The defendant's program utilized the same sample error table as the plaintiff's program and sampled incoming "bits" at the same speed as the plaintiff's programs — choices that did not make functional sense for the defendant's microprocessor. *Id.* at 1494. The defendant's program also contained the same superfluous instructions and the same select call prohibit features as the plaintiff's. *Id.* at 1495–96. Additionally, thirty-eight of the plaintiff's program's forty-four subroutines appeared in the defendant's program. *Id.* at 1496–97.

[316] *Id.* at 1501 n.17.

[317] *Id.* (quoting SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 825 (M.D. Tenn. 1985)). The internal quotations are from *SAS*.

[318] *See supra* notes 77–81 and accompanying text.

[319] 623 F. Supp. at 1501 n.17 (quoting *SAS*, 605 F. Supp. at 825).

[320] *Id.* In another footnote the *Uniden* court stated that the "bottom line" in a computer software context "is whether the copyrighted instructions are the sole means of accomplishing a given task . . . ." *Id.* at 1501 n.16. This statement is correct in the context of an analysis of a merger defense.

The *Whelan*[321] court, in holding that a program's copyright protects a program's structure, obviously recognized that a program's protected expression is not limited to its code. The court noted that it had long been established that works of literature may be infringed without substantial literal similarity. The copyright on a play or book, for example, may be infringed by taking the plot.[322] Thus, the court reasoned that "by analogy to other literary works, it would . . . appear that the copyrights of computer programs can be infringed even absent copying of the literal elements of the program."[323]

In *Whelan*, the court decided the "first impression" question of "whether the structure (or sequence and organization) of a computer program is protectable by copyright, or whether the protection of the copyright law extends only as far as the literal computer code."[324] The plaintiffs in *Whelan* owned "Dentalab," an Event Driven Language (EDL) program helpful in the operation of dental laboratories. The *Whelan* defendants, former sellers of the plaintiff's program, developed and sold a BASIC-language applications program similar to Dentalab.[325] The defendant's program was not a mechanical or direct translation of Dentalab;[326] the differences between EDL and BASIC made a literal translation impossible.[327] As the plaintiff's expert witness testified, there were three similarities between the two programs; the file structures, the screen outputs, and five important "subroutines" virtually were identical in the two programs.[328]

---

[321] Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

[322] *Id.* at 1234.

[323] *Id.* For the viewpoint that the analogy to traditional literary works is inappropriate, see Karjala, *supra* note 36, at 48–51, 61.

Interestingly, the *Whelan* appellate court cited *SAS* as supporting the *Whelan* conclusion that the organization of a program is protected by copyright. According to *Whelan*, the *SAS* court considered "the organizational similarities of the [plaintiff's and defendant's] programs" in reaching its decision that the defendant's program violated the plaintiff's program. 797 F.2d at 1239. *SAS*, it is true, did include a finding that the defendant's program "follows the organizational structure of [the plaintiff's program], down to a detailed level." *See SAS*, 605 F. Supp. at 825–26. One of the plaintiff's experts testified that the defendant's program "employed a very close paraphrase of the [plaintiff's program's] organization and structure." *Id.* at 826. The *SAS* court concluded that the defendant's copying not only affected specific lines of code but also pervaded the entire product, in that it involved copying of the "organization and structural details" of the plaintiff's work. *Id.* at 830. Although this language foreshadows the *Whelan* claim that copying the program's organization is, even without code copying, infringement, the *SAS* defendant apparently copied at a more detailed level than program organization. The *SAS* infringement involved both code similarity and organizational copying. *Whelan* involved only the latter.

[324] 797 F.2d at 1224 (footnote omitted).

[325] *Id.* at 1226. The plaintiff in this case, a software consultant, had originally developed Dentalab for the defendant corporation. *Id.* at 1225–26. The plaintiff then hired the defendant to market the program. The plaintiff's program was written in EDL, the language used in a particular series of IBM computers. One of the individuals associated with the defendant corporation realized that Dentalab, because it was written in EDL, could not be used on the smaller personal computers. He developed a program in the BASIC language and called it "DENTCOM." *Id.*

[326] *Id.* at 1228.

[327] Radcliffe, *supra* note 259, at 196. See *supra* Part III C regarding translations of programs.

[328] 797 F.2d at 1228. One of the defendants' witnesses compared the source code and object code of the two programs and testified about the differences between the programs. That witness concluded that "'substantive differences in programming style, in programming structure, in algorithms and data structures, all indicate that the [defendants' program] is not directly derived from Dentalab.'" *Id.*

The district court held that the defendant's program was substantially similar to the plaintiff's program because the two programs' structures and overall organization were substantially similar.[329] Accordingly, the district court concluded that the defendant's program violated the copyright on Dentalab.[330] On appeal, the *Whelan* defendants contended that the district court's holding of infringement was erroneous because copyright covers only the literal elements of computer programs and the plaintiff did not show that the defendant literally copied the program source code or object code.[331] The structure of a program, the defendants maintained, is by definition the idea and not the expression of the idea.[332] The appellate court, acknowledging that it is frequently difficult to distinguish a work's ideas from the work's expression,[333] formulated "a rule for distinguishing idea from expression in computer programs."[334] The *Whelan* court, citing *Baker v. Selden*,[335] determined that "*the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea.*"[336] Further, the court stated that "where there are various means of achieving the [program's] desired purpose, then the particular means chosen is not necessary to the purpose; hence there is expression, not idea."[337]

This *Whelan* "rule" is, in part, a direct echo of language of the same appellate court's decision in *Apple Computer, Inc. v. Franklin Computer Corp.*[338] In *Franklin*, decided in 1984, the plaintiff Apple alleged that the defendant had duplicated Apple's copyrighted operating systems software. The defendant Franklin admitted duplicating Apple programs.[339] Franklin's goal in copying Apple's operating systems program was to make Franklin's "ACE 100" personal computers compatible[340] with the numerous application programs designed to run on Apple II computers.[341] The district court in *Franklin* denied Apple's motion for a preliminary injunction. The court expressed doubt as to the copyrightability of object code and operating system programs, and held that Apple had not made the necessary showing of likelihood of success on the merits.[342] On appeal, the

---

[329] *Id.*

[330] *Id.* Access was undisputed, the defendant having used and marketed Dentalab. *Id.* at 1225–26.

[331] *Id.* at 1233.

[332] *Id.* at 1235.

[333] *Id.*

[334] *Id.* at 1235–36.

[335] 101 U.S. 99 (1879). See *supra* Parts II C and D for a discussion of *Baker v. Selden*.

[336] 797 F.2d at 1236 (citation omitted, emphasis in original).

[337] *Id.* (footnote omitted).

[338] 714 F.2d 1240, 1253 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984).

[339] *Id.* at 1245. Similarly, in Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984), the defendant, charged with copying Apple operating systems programs to make its "Pineapple" computer Apple-compatible, conceded substantial similarity for the purpose of the appeal. 725 F.2d at 522–23.

[340] See *supra* notes 130–33 and accompanying text for a discussion of "compatibility."

[341] 714 F.2d at 1245, 1253.

[342] Apple Computer, Inc. v. Franklin Computer Corp., 545 F. Supp. 812, 825 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984). Many copyright cases first come to court on the plaintiff's motion for a preliminary injunction. According to "black letter" law on preliminary injunctions, the movant (plaintiff) has the burden of proving four elements: (1) that the movant has a substantial likelihood of success on the merits; (2) that there exists a substantial threat of irreparable injury if the injunction is not issued; (3) that the threatened injury to the moving party outweighs any damage the injunction might cause the defendant; and (4) that

Court of Appeals for the Ninth Circuit held in *Franklin* that a computer program is a "literary work" whether it is in object code or source code, and is thus protected.[343] Next, the *Franklin* court considered whether copyright protection extends to operating systems programs. The defendant proposed two theories that operating systems are not protected by copyright. The defendant contended that operating systems programs either are a process, system or method of operation,[344] excluded from copyright protection by section 102(b) of the Copyright Act,[345] or that operating systems are, in their entirety, unprotected ideas.[346] First, the court disposed of Franklin's contention that the programs were uncopyrightable methods of operation by stating that "Apple does not seek to copyright the *method* which instructs the computer to perform its operating functions but only the instructions themselves."[347] The court analyzed more thoroughly, however, Franklin's contention that an operating systems program is unprotected idea. In so analyzing, the court first noted that "many of the courts which have sought to draw the line between an idea and expression have found difficulty in articulating where [the line] falls."[348] The court then "quoted approvingly" from the 1926 Second Circuit Case *Dymow v. Bolton*:

> Just as a patent affords protection only to the means of reducing an inventive idea to practice, so the copyright law protects the means of expressing an idea; and it is as near the whole truth as generalization can usually reach that, *if the same idea can be expressed in a plurality of totally different manners, a plurality of copyrights may result*, and no infringement will exist.[349]

Dymow does not offer any guidance as to how to separate expression from idea.[350] The *Franklin* court, however, acted as though it did. The *Franklin* court found that guidance

---

the injunction will not disserve the public interest. *Franklin*, 714 F.2d at 1245–46. In a copyright case in which the defendant contends that it took, if anything, only unprotected ideas from the plaintiff's work, analysis of the likelihood of success on the merits will require the court to determine whether the defendant has taken protected expression or an unprotected idea.

As to review of the district court's decision on the motion for a preliminary injunction, appellate courts often have stated that the decision is within the district court's discretion. *See, e.g.*, Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521, 523 (9th Cir. 1984). The appellate court may reverse the district court if the district court applied an incorrect legal standard or abused its discretion. *Id.* at 525.

[343] 714 F.2d at 1249.

[344] *Franklin*, 714 F.2d at 1250–52.

[345] 17 U.S.C. § 102(b) (1982).

[346] 714 F.2d at 1252–53 (citing Baker v. Selden, 101 U.S. 99 (1879)).

[347] *Id.* at 1251 (emphasis added). The appeals court noted that Franklin had persuaded the district court that an operating systems program, unlike an applications program, is part of a machine. *Id.* Franklin also contended that operating systems may not be copyrighted because they are "purely utilitarian works." *Id.* The court of appeals, citing Nimmer as to the proper interpretation of *Baker* concerning works with utilitarian features, refused to accept that argument. *Id.* (citing 1 M. NIMMER, *supra* note 5, § 2.18).

[348] 714 F.2d at 1253 (citing as an example Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930)). The court also noted that the line between idea and expression must be a "pragmatic one," one that keeps in consideration copyright law's balance between preserving competition and protecting authors' works. *Id.*

[349] *Id.* (quoting Dymow v. Bolton, 11 F.2d 690, 691 (2d Cir. 1926) (emphasis added in *Franklin*)).

[350] In *Dymow*, the appeals court, reversing the trial court, held that the defendant's play did not infringe the plaintiff's play by copying a "mere subsection of a plot." 11 F.2d at 692. Each play presented "an ambitious girl of at least potential charm; who is willing to have her ambition served by an ingenious young man, in financial straits." *Id.*

in the *Dymow* language by turning the *Dymow* statement around. From *Dymow's* statement — if the same idea may be expressed in many ways, each expression may qualify for copyright — the *Franklin* court reasoned that if an idea is capable of various modes of expression, any particular mode must be entirely expression.[351] Applying that principle to the *Franklin* facts, the court reasoned that if other programs could be written to perform the same function as Apple's operating systems programs, then the plaintiff's programs necessarily were expression of the idea, not the idea itself.[352]

If the *Franklin* court's holding was that Apple's programs are not, in their entirety, unprotected ideas, the court was correct. If, however, the court's ruling was that the programs are, in their entirety, protected expression, the court was incorrect. Any program contains both unprotected ideas and protected expression. In the discussion that follows the *Franklin* court's ruling that the operating programs are not unprotected ideas, the court appears to have equated unprotected idea with program function. In remanding the case, the *Franklin* appellate court stated that the idea underlying one of Apple's operating systems programs was "how to translate source code into object code."[353] If the unprotected idea is limited to the program's function, as *Franklin* indicated, however, it logically follows that everything contained in the program itself — all lower levels of abstraction beyond function — must be protected expression.

The *Franklin* court's analysis failed to recognize that a program, like a book, contains both expression and unprotected ideas. The literary analogy to the court's reasoning would be to say that *Gone With The Wind* is, in its entirety, protected expression — meaning that every element of the work, including historical setting, geographical setting, and plot outline, as well as character development, literary style, and prose, is protected expression. The *Franklin* defendant, whose goal was Apple-compatibility, viewed the unprotected idea as creating an operating systems program for Franklin computers that would run applications programs designed for the Apple II computer. There were, according to the defendant Franklin, only a limited number of ways of achieving com-

---

[351] *Franklin*, 714 F.2d at 1253.

[352] *Id.*

[353] *Id.* What the court is describing in its example is a compiler, interpreter, or assembler. *See supra* notes 116–20 and accompanying text. The court remanded the case to the district court for factual findings concerning Apple's program but the parties settled.

In *E.F. Johnson Co. v. Uniden Corp.*, discussed *supra* in Part IV, the defendant wanted its logic trunked mobile radios to be "compatible" with the plaintiff's mobile radios, meaning that the defendant's radios could receive signals from and transmit signals to mobile radios made by the plaintiff. The heart of the plaintiff's system was computer software. 623 F. Supp. 1485, 1487 (D. Minn. 1985). Uniden argued that in order to make its mobile radios compatible with the plaintiff's, it was necessary for the defendant to "conform its software program to certain aspects of the plaintiff's mobile radio software." *Id.* at 1501. Uniden further contended that it had to duplicate data tables and certain subroutines in the plaintiff's program in order to achieve that end. *Id.* According to Uniden, the very nature of the idea of achieving the desired system compatibility limits the number of ways the idea may be expressed, making any attempted copyright of the merged expression void *ab initio*. *Id.* The *Uniden* court responded with the *Franklin* court's approach to the identification of idea and expression. The court asked whether other programs could be written or created that perform the same function as the copyrighted program. If they could, the court reasoned, then the program copied is protected expression. *Id.* at 1502. Although the *Franklin* court viewed "the function" as the program's purpose, the *Uniden* court viewed "the function" as the defendant's compatibility objective. *See id.* The *Uniden* defendant's contention failed factually, however, because the verbatim duplication was not necessary to achieve systems compatibility. *Id.*

patibility.[354] This argument is known as the "merger" or "idea/expression unity" defense. Briefly, the "merger" principle provides that if a particular idea may only be expressed in a very limited number of ways, then giving the copyright holder's expression of that idea copyright protection, in effect, gives that copyright owner a forbidden monopoly over the underlying idea itself.[355] The *Franklin* appellate court summarily dismissed this argument, stating the Franklin's goal of compatibility was a "commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[356] If the court had accepted Franklin's conception of the idea as the compatibility goal, Franklin might have been successful in convincing the court that the merger doctrine gave Franklin the right to copy Apple's programs.[357] The characterization of the idea here determines the outcome.[358] Accordingly, the *Franklin* court's view of the unprotected idea in terms of program function doomed the defendant's merger argument.[359]

The *Franklin* appellate court's conclusion that the trial court had erred in denying the plaintiff a preliminary injunction was correct.[360] If a program contains any expression at all, then one who copies "the whole thing" must copy not only unprotected idea, but also the protected expression.[361] The court's characterization of unprotected idea as

---

[354] 714 F.2d at 1253.

[355] Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971). See *supra* notes 82–89 and accompanying text for a discussion of merger.

The *Franklin* court appears to confuse merger with idea identification, stating, in response to the defendant's contention that the plaintiff's programs were unprotected ideas, that its inquiry is whether other programs can be written to perform the same function as Apple's. 714 F.2d at 1253. That *is* the merger question, as the court notes, but it's not a proper question for drawing the line between idea and expression.

[356] 714 F.2d at 1253.

[357] It was unclear, at the time of the litigation, whether compatibility was dependent on using an exact clone of Apple's operating systems programs. *See* Conley and Bryan, *supra* note 36, at 587–91; Grogan and Kump, *supra* note 127, at 115–16.

[358] Nimmer and Krauthaus contend that the separation of expression from idea in copyright cases is merely a mask for the court's policy decision as to the degree of protection to be given the first author and the degree of freedom to be allowed later authors. Nimmer & Krauthaus, *supra* note 36, at 31–32.

[359] In the factually similar case, Apple Computer, Inc. v. Formula Int'l Inc., the district court granted Apple a preliminary injunction. 562 F. Supp. 775, 786 (C.D. Cal. 1983). The defendant appealed, and the Ninth Circuit concluded that it could not hold that the trial court had erred in ruling that Apple had shown a likelihood of success on the merits. 725 F.2d 521, 523 (9th Cir. 1984).

The defendant in *Formula* contended, *inter alia*, that a program is protected under the Copyright Act only if the program embodies expression that is communicated to the user when the program is run. 725 F.2d at 523–24. Operating system programs do not interact with the computer user; they manage the computer. *See supra* notes 127–33 and accompanying text. The appellate court concluded that the 1980 revision of the Copyright Act did not distinguish between programs that interact with the user and those that manage the computer itself. 725 F.2d at 525.

[360] 714 F.2d at 1254.

[361] In theory, the defendant who takes "the whole thing" could still escape a finding of infringement if (1) there are a limited number of ways of expressing the underlying idea; or (2) the plaintiff's program was not original, but was copied from some other programs. *See* E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1499 (D. Minn. 1985).

For other cases in which the defendant copied the plaintiff's entire coded program, see M. Kramer Mfg. Co. v. Andrews, 783 F.2d 421 (4th Cir. 1986) and Midway Mfg. Co. v. Strohon, 564 F. Supp. 741 (N.D. Ill. 1983). In *Strohon*, a videogame case, the plaintiff alleged that the defendant's

program function is, however, a mistake. If the unprotected idea in the plaintiff's program is limited to the overall program function, the plaintiff receives a monopoly over any ideas used within the program. A court's failure to recognize that any program contains many ideas will deprive later program developers of their rightful use of the ideas used in the original program.

Whelan, echoing *Franklin's* application of the idea/expression dichotomy to software, deemed a program's unprotected ideas to be the program's purpose or function plus everything in the program that is necessary to that purpose.[362] In one sense, everything in a program is necessary to the overall program purpose, because everything works toward the overall purpose. The *Whelan* court's meaning of "necessary," however, is more limited than that. In the court's view, "[w]here there are various means of achieving the desired [overall program] purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea."[363] The *Whelan* court, applying its new rule, queried whether the defendants, in copying file structures, screen outputs, and five subroutines, copied unprotected ideas or protected expression. The court stated that "the purpose of [the plaintiff's program] was to aid in the business operations of a dental laboratory."[364] Accordingly, the court analyzed whether anything in the program's structure was necessary to that purpose. Because there were other dental lab operation programs on the market with different structures, the court reasoned that the structure of the plaintiff's program could not be necessary to the purpose.[365] The court distinguished *Synercom* by noting that in *Synercom*, the program's idea and expression could not be identified separately.[366] Thus, by limiting the plaintiff's program's unprotected

---

game violated the plaintiff's program code copyright by copying significant portions of four ROMs containing the instructions that directed the sequence of play. *Id.* at 752. The plaintiff established that 89% of the 16,000 bytes of the plaintiff's program was identically reproduced in the defendant's ROMs. *Id.* The court rejected the defendant's contention that it should, in judging substantial similarity, consider not just the similarity in the instruction ROMs but the plaintiff's and defendant's full programs. *Id.* at 753 (stating that it would "surely be an infringement to copy one chapter of a novel"). Accordingly, the *Strohon* court granted the preliminary injunction. *Id.* at 754.

The defendant in Hubco Data Prods. Corp. v. Management Assistance Inc., 219 U.S.P.Q. (BNA) 450 (D. Idaho 1983), had developed a program for removing the governors that restricted the plaintiff's operating systems program to below-peak levels of operation. The defendant's method required copying the plaintiff's entire object code. The court granted a preliminary injunction.

[362] *See Whelan*, 797 F.2d at 1236.

[363] *Id.* (footnote omitted). The court noted and rejected the arguments against extending copyright protection beyond protection for code copying. *Id.* at 1237–38.

The *Whelan* court viewed its rule as striking the correct balance — rewarding innovation without giving innovators "a stranglehold over the development of new computer devices that accomplish the same end." *Id.* at 1237.

The court indicated that copying of *scenes a faire*, although expression, is permitted because the subject matter expressed through a *scene a faire* can be expressed no other way. *Id.* at 1236. *See supra* notes 90–92 and accompanying text. No case yet has considered whether there are software structures, sub-routines, or code phrases that parallel *scenes a faire* in literary work. The courts *have* recognized *scenes a faire* in the videogame context. *See, e.g.*, Atari, Inc. v. North Am. Philip Consumer Elec. Corp., 672 F.2d 607, 616 (7th Cir.), *cert. denied*, 459 U.S. 880 (1982).

[364] 797 F.2d at 1238. The court noted that its rule might not apply to non-utilitarian works, the purpose or function of which cannot be stated. *Id.*

[365] *Id.*

[366] *Id.* at 1240. The *Whelan* court disposed of *Synercom* further by noting that the copyright statute now specifically extends copyright protection to "compilations" and "derivative works". *Id.*

idea to the program's purpose, the *Whelan* court concluded that the program's structure was expression rather than idea.[367]

The *Whelan* court's "rule" is a blunt instrument. Under this rule, virtually nothing may lawfully be borrowed from a copyrighted program; the rule limits the unprotected idea to program function. Although the *Whelan* formula for the unprotected idea includes, in addition to overall program purpose, all that is "necessary" to that purpose, the court answers the question of whether program content is necessary to the program purpose by asking whether there are dissimilar programs that fulfill the same broad purpose. If there are, the plaintiff's work's unprotected idea is limited to the program purpose. A completed program necessarily contains many ideas, however, for ideas must be applied to solve the program's subproblems.[368] Some levels of detailed problem-solving beyond program purpose must be unprotected ideas. The question is at what point between overall program purpose, the highest level of abstraction, and actual code, the lowest level of abstraction, should a line be drawn. Accordingly, to say that nothing may be taken from a program violates copyright law's axiom that copyright does not protect the ideas used in a work. If everything beyond overall program purpose is protected expression, copyright holders get more protection than they should.[369]

The *Whelan* "rule" for distinguishing idea from expression may, of course, be adjusted. If a lower court wishes to pay lip service to the *Whelan* rule, yet still reach the conclusion that a defendant who has duplicated more than the plaintiff's program's overall function has not infringed, the court may either define program purpose or function more specifically or decide that what the defendant copied *was* necessary to achieve the purpose. In a footnote, the *Whelan* court may have left an opening for such interpretations, when the court stated that it "[did] not mean to imply that the idea or purpose behind *every* utilitarian or functional work [would] be precisely what it accom-

---

at 1239. The court noted that a "compilation" is defined as "a work formed by the collection and *assembling* of preexisting materials or of data that are selected, *coordinated, or arranged* in such a way that the resulting work as a whole constitutes an original work of authorship . . ." *Id.* (quoting part of 17 U.S.C. § 101 (emphasis original to the case but not the statute)). Further, the court stated that a "derivative work" is a work "based upon one or more preexisting works, such as . . . [an] *abridgement, condensation* or any other form in which a work may be *recast,* transformed, or adapted." *Id.* The *Whelan* court concluded that Congress "intended sequencing and ordering [of data] to be protectible in the appropriate circumstances." *Id.* at 1240.

[367] *Id.* at 1238–39. The *Whelan* district court had also equated the unprotected idea with program function, stating as follows:

> [T]he mere idea or concept of a computerized program for operating a dental laboratory would not in and of itself be subject to copyright . . . . The expression of the idea . . . is the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating and producing useful information.

Id. (quoting 609 F. Supp. 1307, 1320 (E.D. Pa. 1985)).

[368] *See supra* Parts III B and IV A.

[369] The *Whelan* court should have realized that a program contains many unprotected ideas, for the court began its analysis by describing the various steps involved in a program's creation. The court noted that program design includes: (1) identifying the problem (in this case, record-keeping for a dental laboratory); (2) developing an outline or "flow chart" breaking the solution down into a series of smaller units; (3) creating data files to govern the arrangement of data input; and (4) translating each step in the detailed design into a computer language such as EDL or BASIC. 797 F.2d at 1229–30. The court did not, however, use these design concepts in developing its "rule". *See id.* at 1236.

plishes, and that structure and organization [would] therefore always be part of the expression of such works."[370] The court acknowledged that the idea or purpose underlying a utilitarian work could be "to accomplish a certain function *in a certain way*."[371] In that situation the structure of the program might be essential to that specific task, and so be unprotected ideas.[372]

Broderbund Software, Inc. v. Unison World, Inc.,[373] decided a few months after *Whelan*, may be the first of many cases to follow *Whelan's* approach of equating idea and program function. In *Broderbund*, Judge Orrick ruled that the separable idea of the plaintiffs' program "Print Shop" was to use computers to create greeting cards, banners, posters, and signs.[374] Judge Orrick held that, although other software publishers are free to market programs designed to let the user create greeting cards, banners, posters and signs, their expression of that idea must, to be non-infringing, be made "through a substantially different structure" than that used by the plaintiffs.[375] Judge Orrick, comparing the plaintiff's "Print Shop" with the defendant's program, concluded that the defendant's program "looks like a copy of 'Print Shop', with a few embellishments scattered about in no particular order."[376] Judge Orrick held that the two programs' sequences of screens, layout of screens, and method of user feedback were substantially similar, and accordingly, ruled that the defendants infringed the plaintiff's copyright.[377]

In *Broderbund*, the plaintiff did not claim that the defendant copied the code or even the structure of the plaintiff's program. Instead, the plaintiff contended that "the overall appearance, structure, and sequence of the audiovisual displays" of the defendant's program infringed the plaintiff's copyright.[378] The defendant argued, however, that the

---

[370] *Id.* at 1238 n.34.

[371] *Id.*

[372] *Id.*

[373] 648 F. Supp. 1127 (N.D. Cal. 1986).

[374] *Id.* at 1133 (noting *Whelan's* identification of Dentalab's idea, efficient organization of a dental lab).

[375] *Id.*

[376] *Id.* at 1137. The plaintiffs' "Print Shop" program operated only on Apple computers. The defendant, Unison, specialized in converting other publishers' software to make it adaptable to different computers. The plaintiffs, desiring an IBM version of "Print Shop," began discussing with the defendant the possibility of having Unison produce an exact reproduction of the original program for IBM computers. One of the defendant's programmers then was directed to develop a program as identical to "Print Shop" as possible. *Id.* at 1130–31. The programmer who created the program for the plaintiff "very briefly showed the source code" to one of the defendant's programmers." *Id.* Unison also had commercially available copies of the program. *Id.*

Unison's programmer had made considerable progress towards a reproduction of "Print Shop" when negotiations between plaintiff and Unison broke down. The defendant's president instructed the programmer to "stop copying 'Print Shop' and finish developing an enhanced version of the greeting card program." *Id.* at 1131. By this time, however, the programmer already had finished the menu screens and ten screens in the "greeting card" and "sign" functions. *Id.* When Unison began marketing its program, the plaintiff claimed that "the overall appearance, structure, and sequence of the audiovisual displays" in the defendant's program infringed the plaintiffs' copyright. *Id.* at 1130.

[377] *Id.* at 1137.

[378] *Id.* at 1130. Judge Orrick was incorrect when he stated that *Whelan* "stands for the proposition that copyright protection is not limited to the literal aspects of a computer program, but rather that it extends to the overall structure of a program, including its audio-visual displays." *Id.* at 1133. The first part of that sentence is correct. The second clause, however, is inaccurate. *See Whelan*, 797 F.2d at 1244.

idea underlying the menu screens, input formats, and screen sequencing was indistinguishable from the expression, because any menu-driven program that allows users to print cards and signs must have a user interface like that of "Print Shop."[379] The *Broderbund* court, following *Whelan*, identified the overall program's function — create greeting cards — as the unprotected idea,[380] and held that other expressions of the function were not only possible, they were on the market.[381] The *Broderbund* court apparently viewed the menu screens and sequence of screens as entirely expression, on the grounds that they are neither the overall program function nor essential to that function.

The *Broderbund* analysis contrasts sharply with *DCA's* analysis of a factually-similar alleged infringement. In both cases the alleged infringement involved copying user interface features rather than copying code. In *Broderbund* the court, adopting *Whelan's* rule for separating idea from expression, equated overall program function with unprotected idea.[382] In *DCA*, however, the court "looked inside" the user interface features of the two programs and characterized individual features as either idea or expression.[383] Although *Broderbund's* holding of infringement appears correct given the extent of the defendant's taking, *Broderbund's* idea/expression analysis based on *Whelan* is worrisome.

V. The Impact of the *Whelan* Rule

The separation of a work's ideas from its expression amounts to a judgment about the degree of freedom to be given to subsequent authors and the degree of protection to be given to the first author.[384] The copyright law axiom holds that expression is protected, ideas are not. Identifying as expression that which the defendant took is, for practical purposes, a conclusion of infringement. Nearly thirty years ago Judge Learned Hand stated that "no principle can be stated as to when an imitator has gone beyond the 'idea,' and has borrowed [a work's] 'expression.' Decisions must therefore inevitably

---

[379] 648 F. Supp. at 1132.

[380] *Id.* at 1133.

[381] *Id.* at 1132. The court quickly disposed of the defendant's merger argument, noting that evidence at trial indicated that there were other programs that allowed users to do what the plaintiffs' program did — print greeting cards, signs, banners, and posters. The court referred to one such program, "Sticky Bear Printer," and determined that though the ideas of "Print Shop" and "Sticky Bear Printer" are the same, the expressions of those ideas, the menu screens and sequence of screens, were different. *Id.* The presence of a same-function program with a different user interface structure was, in Judge Orrick's view, sufficient to disprove the defendant's merger contention. *Id.*

[382] *Id.* at 1133.

[383] *See* 659 F. Supp. 449, 459 (N.D. Ga. 1987).

DCA and *Broderbund* also diverge on whether a program's copyright extends to the audiovisual display. *Broderbund* held that copyright does extend to audio visual displays. 648 F. Supp. at 1133 (interpreting *Whelan*: "*Whelan* thus stands for the proposition that copyright protection is not limited to the literal aspects of a computer program, but rather that it extends to the overall structure of a program, including its audiovisual displays."). In contrast, the *DCA* court held that the program copyright does not extend to the screen display. 659 F. Supp. at 455. The *DCA* court opined that the *Broderbund* court erred by reading *Whelan* too expansively. *Id.* According to *DCA*, *Broderbund's* conclusion on this question rested on "an over expansive and erroneous reading of *Whelan*." *Id.*

[384] Nimmer and Krauthaus, *supra* note 36, at 31. Those authors view all judgments defining a work's ideas and its expression as "inherently unstable." *Id.* All aspects of a work, they say, "entwine idea *and* expression." What constitutes expression and what constitutes an idea for *purpose of infringement* reflects a judgment about the work and about whether protection should be granted against a specific subsequent product." *Id.* (emphasis in original).

be *ad hoc*."[385] More recently commentators have criticized the idea/expression dichotomy as lending itself to results-oriented decisions.[386] Concern about results-oriented judicial decisions is particularly well-founded in software copyright idea/expression cases, for it is common in these cases for the plaintiff to have gone to great expense to develop and market its software.[387] To the extent that a judge is horrified by the "free ride" that a defendant may get by taking some aspects of the plaintiff's expensively-developed software or by mimicking the software's popular features, it is easy for a judge to conclude, under an *ad hoc* approach, that what was taken by the defendant was protected expression.

That the *Whelan* court desired to create a rule for separating a program's ideas from its expression is understandable. The rule stated in *Whelan*, though, is too blunt an instrument. The *Whelan* court's rule makes everything in a copyrighted program protected expression. If "purpose" is defined as broadly as it was in *Whelan* — "aid in the business operations of a dental lab" — there will always be various means of achieving that purpose. One program might aid in the business operations of a dental laboratory by handling orders; another might aid in the business operations of a dental laboratory by automating billing; yet another program might achieve that broadly-stated purpose through inventory control. None of the internal program content in any one of these programs is necessary to the overall purpose, for the other two programs are alternate means of aiding in the business operations of a dental laboratory. Under *Whelan's* rule, then, each entire program — each "particular means chosen" — must be protected expression rather than unprotected idea.[388]

In contrast to *Whelan's* rule, the principle of merger expands the range of existing material that may lawfully be used by later authors. Merger permits second and later authors to use a first author's expression when it is necessary to use that expression to use the underlying idea.[389] Where the first author's expression is inseparable from the ideas expressed, that expression may lawfully be used by others, for to protect the expression would give the first author a monopoly on the idea. *Whelan* twists the merger principle around: though merger makes unprotectable the expression that is necessary for use of the underlying idea, *Whelan's* rule extends protection to all that is not necessary for a broadly-defined desired purpose. As another writer has noted, "[s]omething could both be unnecessary to achieve that [broadly-stated] purpose and be an idea that the copyright laws do not (and should not) permit anyone to own."[390] While merger prohibits idea monopolization, the *Whelan* rule, by making an entire "means chosen" protected expression, permits idea monopolization.

Whelan's failure to recognize that program function and unprotected ideas are not synonymous closes off from general use the ideas put to work in a protected program.

---

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

[386] *See supra* notes 102–07 and accompanying text. One court has noted the criticism, but characterized it as criticism "more of the application of the distinction than of the distinction itself." *See* Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1163 n.6 (9th Cir. 1977).

[387] *See, e.g.*, Synercom Tech. Inc. v. University Computing Co., 462 F. Supp. 1003, 1008 (N.D. Tex. 1978).

[388] *See* Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1236 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

[389] *See* Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971).

[390] Stern, *MicroLaw: Software Copyright Developments*, IEEE MICRO, 74, 75 (Dec., 1986).

While the authors of the CONTU report envisioned a world in which "programmers are free to read copyright programs and use the ideas embodied in them in preparing their own works,"[391] the *Whelan* rule makes any use of a pre-existing program risky.[392] Appropriately, *Whelan's* rule makes infringing the program prepared from an existing program by the "clever copier" who uses a text-editor to make trivial changes in an existing program. That rule also, however, makes infringing any translation of an existing program, even the translation accomplished with great added effort. Any translation of a program embodying "non-necessary" features of the original program is, under *Whelan's* rule, a taking of protected expression. The programmer whose earlier works are copyright-protected by his former employer will not, under the *Whelan* rule, be able to write programs similar even in structure and organization to his old programs. User interface copying is, under *Whelan's* rule, infringement if there is another way of doing whatever was being done in the first program.[393]

Whelan's rule for separating ideas from expression will, if followed, create a lopsided intellectual property system that over-protects early developers at the expense of later developers. Suppose a "Developer 1" makes minor additions to a program that long has been in common use among programmers. Because only minimal originality is required for a "fixed" work to qualify for copyright protection,[394] Developer 1's program is protected. If Developer 1's protection extends to all in that program that is not necessary to the overall program function, it appears that Developer 1 now has a years-long intellectual property monopoly on the entire program, including the part of the program he or she took from the public domain.[395]

In the software industry, it once was thought that "clean-room" procedures could be used to ensure that a later program, "Program 2," based on some use of an earlier program, "Program 1," did not infringe Program 1. In a clean-room procedure, two teams of programmers are set up to handle two separate tasks to create an emulator of

---

[391] CONTU Report, *supra* note 33, at 40–41.

[392] Suppose that Programmer #2, who wishes to create a spell-check program, studies an existing spell-check program ("Program 1") which breaks the user-created document into words, compares each word with a dictionary, and prints non-matches. Under *Whelan*, Programmer #2 is free to create a spell-check program, and he is free to use anything from Program 1 that is necessary to the function "check spelling." Presumably breaking the document into words is necessary for that function, as is using a dictionary comparison step. Suppose that Program 1 displays the non-matches after each individual word is entered. The program could, however, display non-matches at the document's end. Is Programmer #2 prohibited from choosing the "display non-matches" route used in Program 1? Program 1's route is not necessary to the overall function "check spelling" or even to the subfunction "display non-matches." Under *Whelan*, the display manner used in Program 1 appears to be protected expression.

[393] One commentator recently referred to the judicial trend in software copyright cases as "the new protectionism." Karjala, *supra* note 36. The broad protection that *Whelan's* rule gives the first author's program is contrary to recent recommendations by Karjala and other commentators that the level of copyright protection given to programs be limited. *See* Goldstein, *supra* note 36; Menell, *supra* note 36, Nimmer & Krauthaus, *supra* note 36.

[394] *E.g.*, Synercom Tech. v. University Computing Co., 462 F. Supp. 1003, 1009–10 (N.D. Tex. 1978).

[395] A somewhat analagous situation arises with respect to derivative works. According to the copyright statute's section on derivative works, the copyright in a derivative work does not extend to preexisting material not contributed by the author of the derivative work. 17 U.S.C. § 103(b) (1982). Perhaps section 103(b)'s principle would be applied to the situation described in the text to which this footnote is appended.

an existing program. One team analyzes the existing target software, learns how it works, and then delivers specifications to the second team. The programmers on the second team then write program code based on those specifications. The second team members have no access to the original program's code, and the first team members have no part in writing the new code. Prior to *Whelan*, it was thought that the resulting new code of "Program 2" would embody only the ideas used in the first program.[396] If, though, the "idea" is limited to overall program function and whatever is necessary to the broadly-stated function, much of what is passed on to team #2 in specifications is expression.[397]

The clean-room procedure is used not just by those who want to emulate the features of popular software, but also by those desiring to achieve compatibility with some popular computer. The compatible computer must have operating systems software that mimics the operating systems software used in the chosen model. For example, to make an IBM-compatible computer a developer must provide a basic input/output system (BIOS). The compatible computer's BIOS is the part of the operating system that interfaces between the user's applications programs and the hardware, producing the same functional results as the IBM BIOS for "interrupt" calls to given interrupt numbers. Under *Whelan's* rule the interrupt numbers and parameters would be expression because the numbering chosen by IBM and the parameters chosen are not necessary to the overall function of serving as an interface between user programs and hardware.[398] Additionally, the clone-maker's emulator BIOS must have modules and subroutines that mimic the modules and subroutines in the IBM BIOS. Under *Whelan*, however, the IBM BIOS module/subroutine structure is protected expression rather than unprotected idea so long as that particular structure is not necessary to the overall "interface with applications programs" function of the BIOS. A similarly-structured BIOS, even if prepared through clean-room procedures, would then be infringing, making it impossible to achieve compatibility without infringing a copyright.[399]

There are two obvious cures for the over-protection that results from the *Whelan* error of extending copyright protection beyond code without "looking inside" the protected program for unprotected ideas. First, courts could restrict software copyright protection to literal code duplication. The second cure for *Whelan's* mistake is to extend copyright protection for software beyond literal code, as *Whelan* did, but to use the Hand-Nimmer "levels of abstractions" test to determine, on a case-by-case basis, at what

---

[396] Stern, *supra* note 390, at 76–77; *see* Davis, *supra* note 127; Reback & Siegel, *supra* note 134 (applying "clean room" concept to goal of developing an IBM-PC compatible computer).

[397] To go back to the "spell-check" example used in Part III B, if team 2 is told to develop a spell-check program that displays non-matches after each word is entered, the route used in Program 1, then expression is borrowed because there is an alternative — the non-matches could be displayed at the end of the document entry.

[398] Reback & Hayes, *supra* note 204, at 6. Reback and Hayes give other examples of *Whelan's* long reach.

[399] There are four possible solutions to the compatibility foreclosure problem. First, courts could make operating systems software non-copyrightable. *See* Menell, *supra* note 36; *cf.* Goldstein, *supra* note 36, at 1126–30 (suggesting that fair use and "copyright misuse" doctrines may provide a solution to this problem). Second, courts could treat organization copying done for compatibility purposes as non-infringing fair use. A third solution might be to require that copyright holders grant licenses to those who must mimic operating systems programs for compatibility purposes. Finally, if program "purpose" is the unprotected idea, define the purpose of these programs in terms of compatibility.

level of abstraction between overall program function and actual code protection should begin. Other commentators already have suggested that because software is created under top-down program design principles,[400] the Hand-Nimmer abstractions test is particularly appropriate for software.[401] *Uniden* also suggests that approach.[402]

In applying the abstractions test to software, the code is the lowest level of abstraction; the overall program function — "check spelling," or "assist in the efficient operation of a dental laboratory" — is the highest level of abstraction. For a spell-check program the second highest level of abstraction would appear to be the subfunctions picked for that program — "divide user document into words," "check words in dictionary," and "display non-matches." The modules and subroutines are more abstract than the code but less abstract than the overall function. The modules may be dictated by the overall function and thus ideas, or they may be expression. The same is true of subroutines. At this level the functionally-suggested modular structure should be open for use by all as should common or "libraried" subroutines, just as in literature, under the *scenes a faire* doctrine, stock or standard characters may be used by later writers.[403] A functionally-*required* modular or subroutine structure would be unprotected either as an idea or under the merger doctrine.[404] Perhaps the *Plains* appellate court was suggesting such an approach for that case. The court, affirming the lower court's denial of a preliminary injunction, stated that while it saw similarities between plaintiff's and defendant's works, the record supported an inference that market factors were significant in determining the sequence and organization of cotton marketing software.[405] The court declined to hold that those patterns could not be ideas in a computer context.[406]

Drawing the line between a program's idea and its expression is, as critics of the idea/expression analysis have said, ultimately a policy judgment on the scope of protection.[407] The idea/expression line drawing is meant to be used to balance the right of the copyright holder with the rights of other creators.[408] The level-of-abstractions analysis is the most principled manner of balancing these interests.

Section 102(b) of the Copyright Act not only makes the ideas in a copyrighted work unprotected, it also makes processes and methods unprotected. The *Whelan* court did not consider that protection of program structure could violate section 102(b)'s prohibition against protecting processes or methods.[409] If the unprotected idea is limited to the overall program function, as in *Whelan*, the copyright holder receives a monopoly on

---

[400] *See supra* Part III B.

[401] *See supra* notes 203–04 and accompanying text.

[402] *See* E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1501 n.17 (D. Minn. 1985). See *supra* Part IV B for a discussion of *Uniden*.

[403] See *supra* notes 90–92 and accompanying text.

[404] See *supra* notes 82–89 and accompanying text.

[405] 807 F.2d at 1262.

[406] *Id.*

[407] *See supra* notes 102–07 and accompanying text.

[408] *See* Chafee, *supra* note 56, at 511–14.

[409] In *Whelan* the Third Circuit adopted functional, "process-like" language to describe expression, approvingly quoting the trial court's definition of expression as "the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information either on a screen, print-out or by audio communication." 797 F.2d at 1239.

the process or method. Use of the Hand-Nimmer levels of abstractions test on a case-by-case basis would allow courts to leave "the process" unprotected.[410]

Applying an abstractions test to a program *is* more difficult than applying the *Whelan* rule — the court must "look inside" the program to apply the abstractions test. Courts cannot apply the abstractions test to software without help from experts. If the plaintiff's program is written in one programming language and the defendant's in another, use of an expert is necessary to show to what extent the defendant's work tracks the plaintiff's work. The software copyright cases have recognized the necessity of using experts.[411]

VI. *Conclusion*

Now that it is clear that copyright protects computer programs, courts will be called upon to decide what, in a given program, is protected expression and what is unprotected idea. The separation of a program's expression from its ideas should be done on a case-by-case basis. Protection should not be restricted to protection against literal code duplication or code paraphrasing, for it is easy to make a program look different from one that was copied. Beyond that, because coding is the last step in developing a program,

---

[410] *Cf.* Karam, *supra* note 36, at 30–33 (slightly different approach to the use of section 102(b)'s prohibition on process protection in software infringement cases).

[411] *See Whelan*, 797 F.2d at 1233; SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 818 (M.D. Tenn. 1985).

Generally the courts have stated that the factual question of substantial similarity in copyright cases should be answered by applying an "ordinary observer's" reaction to the two works. 3 M. NIMMER, *supra* note 5, § 13.03[E]. According to the "lay observer" or "audience reaction" test, an "ordinary person" should be able to detect the defendant's literary piracy without any suggestion or critical analysis by others. *Id.* Some courts have interpreted this statement as prohibiting the use of expert testimony in the factual determination of substantial similarity. Nimmer criticizes this approach, stating that there are numerous instances where the "ordinary observer" cannot detect appropriation. *Id.* § 13.03[E], at 13-52. Nimmer argues that there is no reason to disallow suggesting and pointing out similarity if the suggestion will help the trier of fact see that the plaintiff's work formed the basis for defendant's work. *Id.*

Modifications of the "ordinary observer" test have been developed by the federal Courts of Appeal for the Second Circuit and Ninth Circuit, where many of the major copyright cases are filed. In *Arnstein v. Porter*, a music copying case, the Second Circuit bifurcated the question of substantial similarity into two sub-questions. On the first question, whether the defendant copied from plaintiff's work, both protectible and non-protectible (i.e., idea-level) elements of the works are compared; expert analysis and dissection are permitted. Once copying has been established, the question of whether the defendant's appropriation is unlawful must then be decided according to the "ordinary observer" test. 154 F.2d 464, 468–69 (2d Cir. 1946).

Over thirty years later, the Ninth Circuit developed its own application of *Arnstein's* bifurcation. *See* Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1164 (9th Cir. 1977). The *Krofft* court recognized that similarity of expression, not just ideas, must be present for a copyright infringement to exist. As a starting point, however, the court must determine whether the plaintiff's and defendant's works have even idea-level similarities. That determination, called "extrinsic" in *Krofft*, may be based on specific criteria and with the help of analytic dissection and expert testimony. *Id.* If the works have idea-level similarities, the question of whether there is substantial similarity between the two works at the level of protected expression (the "intrinsic" question) must be made on an *ad hoc* basis by the trier of fact using the responses of an ordinary reasonable person. *Id.* The "extrinsic" determination may be made as a matter of law on a motion for summary judgment. *See* Frybarger v. International Business Mach. Corp., 812 F.2d 525, 528 (9th Cir. 1987).

a program's expression begins somewhere between the overall program function and the code.

If "expression" is something more than code — if a copyright holder is protected from non-literal use of his work — any program is likely to contain many unprotected ideas. Courts can, by using the traditional "level of abstraction" approach, draw the line between what is free for use by others and what is protected.