Brooklyn Law Review

Volume 59
Issue 2
The Second Circuit Review: 1991-1992 Term

Article 5

2-1-1993

# *Computer Associates International v. Altai, Inc.*: Protecting the Structure of Computer Software in the Second Circuit

Andrew Isztwan

# *COMPUTER ASSOCIATES INTERNATIONAL v. ALTAI, INC.*:[*] PROTECTING THE STRUCTURE OF COMPUTER SOFTWARE IN THE SECOND CIRCUIT

## INTRODUCTION

Copyright protection for computers, like the evolving technology of computers itself, is in a state of flux. Although Congress decided in 1980 that computer software should be protected by copyright law instead of patent law,[1] the propriety of that decision is still being questioned by courts and commentators.[2] The Second Circuit recently narrowed the scope of computer software copyright protection in *Computer Associates International v. Altai, Inc.*[3] While this decision may restrain the growth of copyright as a barrier to the reuse of software, it fails to provide critically needed, specific guidance in the evidentiary analysis of copyright infringement in computer program structure.

Computer programs are implicitly included in copyrightable subject matter,[4] and courts have had little or no trouble granting copyright protection to what commonly have been referred to as the "literal" elements of computer software.[5]

[*] 982 F.2d 693 (2d Cir. 1992).

[1] *See* FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS (1978), *reprinted in* 5 COPYRIGHT, CONGRESS AND TECHNOLOGY: THE PUBLIC RECORD (Nicholas Henry ed. 1980) [hereinafter CONTU FINAL REPORT]. Patent law protects ideas or processes while copyright law protects only the particular expression of an idea. Patent law protection is more extensive in scope and, therefore, entails a more rigorous examination process and provides a shorter duration of protection than copyright.

[2] *See* Bradford P. Lyerla, *Copyrightability of Software User Interfaces: The Natural Law Versus the Social Utilitarian Approach*, COMPUTER LAW., Jan. 1993, at 21 (comparing the views of those approving of copyright protection for software with those abhorring it); Marc T. Kretschmer, *Copyright Protection for Software Architecture: Just Say No!*, 1988 COLUM. BUS. L. REV. 823 (advocating reliance on carefully restricted access to software production in conjunction with trade secret law protection).

[3] 982 F.2d 693 (2d Cir. 1992).

[4] Computer programs are treated as "literary works" under 17 U.S.C. § 102(a) (1977). H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 (1976).

[5] *See* Apple Computer v. Franklin Computer, 714 F.2d 1240 (3d Cir. 1983),

Literal elements include the sequence of computer instructions that are actually "written" by the programmer and fixed in a form readable by either people or computers. Thus, such "writings" can easily be viewed as "original works of authorship fixed in [a] tangible medium" as required for protection under the Copyright Act of 1976.[6]

On the other hand, courts have wrestled with the copyrightability of so-called "nonliteral" elements of computer software. Nonliteral elements include the organization, structures and dynamic sequences of a "running" program.[7] While nonliteral elements of traditionally copyrightable material are protectible, courts must always face the difficult question of where to mark the boundary between unprotectible "ideas" and the protectible "expression" of an idea.[8] This problem is compounded when applied to computer software for two reasons. First, the nonliteral "expressions" (specifically, the structures) of a computer program are not easily seen or "felt" as may be the nonliteral expressions of a play, for instance. Second, the fact-finder typically has no computer training and, thus, has no ability to identify these expressions or to determine whether they are similar.

The Second Circuit addressed both of these problems in *Computer Associates*. Initially, the court created a three-part test that can be applied to any software to determine whether its structure is protectible.[9] Next, in addition to considering the testimony of expert witnesses brought forth by the parties, the court appointed its own expert witness to guide it through the intricacies of computer software design and to render significant aid in deciding the substantial similarity issue.

The Second Circuit's three-part test is cognizant of the realities of computer programming. Yet the court failed to offer useful guidance to its application in future cases. Further, the court accepted that expert testimony is necessary in any com-

*cert. denied*, 464 U.S. 1033 (1984); Williams Elecs. v. Artic Int'l, 685 F.2d 870 (3d Cir. 1982); CMS Software Design Sys. v. Info Designs, 785 F.2d 1246 (5th Cir. 1986).

[6] 17 U.S.C. § 102(a) (1992).

[7] A computer program that is in operation is said to be "running."

[8] *See* Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930) (Hand, J.) (discussing this distinction).

[9] Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 706-11 (2d Cir. 1992).

puter software litigation because the fact-finders typically have no experience in the field. In doing so, however, the Second Circuit dangerously allowed a court-appointed expert's testimony to cross the boundary between assisting the fact-finder and creating law.

Part I of this Comment describes the basic computer software concepts necessary to an understanding of the underlying material, and surveys the development of copyright law as it applies to software. Part II next explores the *Computer Associates* opinions of the district court and Second Circuit. Part III then analyzes the "literal" and "nonliteral" distinction, elaborates on the concept of computer program structure and examines the Second Circuit's test for substantial similarity of program structure, and considers the extent to and manner in which expert testimony should enter the analysis. Finally, this Comment suggests how courts should explain their reasoning in computer software cases in order to improve the clarity and precedential value of their decisions.

## I. BACKGROUND

### A. *Basic Computer Concepts*

"Computer programs" are defined in the Copyright Act of 1976 as "set[s] of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[10] These instructions are actually "written" by a computer programmer, who understands a language of special words and symbols that can be reduced to a format intelligible to a computer. Computer programs are called "software" because they are malleable: a programmer can easily change a set of instructions to the computer simply by editing what has been written.

The computer itself, as that term is used in the Copyright Act, is considered "hardware." Hardware consists of the physical electronic circuits in which the processing ordered by the instructions occurs. The term "hardware" is appropriate since these components are relatively fixed: a program can order the circuits to perform certain predefined functions, but the config-

[10] 17 U.S.C. § 101 (Supp. I 1993) (amending 17 U.S.C. § 101 (1976)).

uration in which the circuits are arranged inside the computer is set during the manufacturing of the computer and cannot easily be altered thereafter.

Computers only understand instructions that are represented as sequences of the binary digits zero and one. Writing a program as an endless sequence of binary numbers would be impossible, however, so programs are ordinarily written in one of many programming languages understandable by humans.[11] A program in this form is known as the "source code." Once a program is completely written, it must be translated to be understood by the computer. A computer program known as a "compiler" translates the source code into a form known as the "object code," the computer-legible sequences of zeroes and ones. Hence, two entirely different collections of symbols which specify exactly the same set of instructions exist for each program: the source code for the programmer and the object code for the computer.[12]

Computer hardware is complex and can be difficult to coordinate. Disk drives, monitors, keyboards and memory all require special sequences of instructions to make them run correctly and efficiently.[13] To control the hardware directly, a programmer needs to know about every component of the system to a level of detail nearly approaching the configuration of all the millions of electronic circuits. To ease the burden on

---

[11] Like human languages, a programming language possesses both a vocabulary (a fixed set of instructions represented by sequences of letters or symbols) and a grammar (a set of rules for combining the elements of the vocabulary). The vocabularies of these languages are typically sets of abbreviations or mnemonics for words or phrases of a human language.

[12] There may, in fact, be more than two collections of symbols for a given program. Some compilers accomplish the transformation from source to object codes in more than one step and generate temporary translations in intermediary languages. Because both source and object codes are widely held to be copyrightable, *see infra* note 28 and accompanying text, it is likely that these intermediate translations are also copyrightable.

[13] Although both are technically forms of memory, a "disk drive" typically refers to a permanent storage device while "memory" usually refers to temporary storage (e.g., Random Access Memory, or RAM). The object code of a computer program is held in permanent storage; when it is run, it is copied into temporary storage where the computer can actually read the program and process data.

Because a computer program must be copied from one storage medium to another in order to function, the Copyright Act of 1976 was amended in 1980 to exclude this form of copying from constituting infringement. 17 U.S.C. § 117(1), *amended by* Pub. L. 96-517, § 10(b), 94 Stat. 3028 (1980).

programmers, and to maximize the efficient use of the computer's hardware resources, a special program known as an "operating system" is present in most computer systems. An operating system is a sort of "command program:" it manages and efficiently distributes the hardware resources of a computer (permanent and temporary memory, access to various processing units, etc.) to all of the other programs that are running. In addition, programmers creating applications for a computer can write their software to meet the specifications of the operating system, which is far easier than writing to meet the more complex specifications of the hardware of the computer itself.[14] An operating system thus "hides" the most difficult chores from the application programmer. Yet each operating system generally has its own unique set of specifications that an application program must meet. Thus, a program is written to run only on a specific computer machine, and in conjunction with a specific operating system. Software created for one combination of hardware and operating system typically will not function with a different combination.[15]

## B. *The Development of Copyright Law for Computer Software*

The Copyright Act of 1976 gives authors the exclusive rights to reproduce, distribute, display and perform their works, as well as the right to prepare derivative works based on their copyrighted works.[16] These rights endure for the life of the author and fifty years after the author's death.[17] Copyright protection is limited to "original works of authorship fixed in any tangible medium of expression."[18] Copyright protects only the *expression* of ideas; it does not extend to ideas,

[14] An application is a program intended to be operated by a user and not another programmer. That is, it is meant to be of use to people who wish to take advantage of the capabilities of a computer without having to know in detail how the computer works. A common example of an application program is a word processor.

[15] Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 698-99 (2d Cir. 1992).

[16] 17 U.S.C. § 106 (1992).

[17] 17 U.S.C. § 302(a) (1992).

[18] 17 U.S.C. § 102(a) (1992). The tangible medium of expression may be "now known or later developed, from which [the work] can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device." *Id.*

procedures, processes or methods, which can be protected only by patent law.[19]

Although the Copyright Office has permitted authors to register computer programs since 1964,[20] doubt as to the validity of protection for computer programs under the Copyright Act of 1909 prompted Congress in 1974 to create the National Commission on New Technological Uses of Copyrighted Works ("CONTU") to study copyright issues of computer programs and photocopying.[21]

In 1978, CONTU issued its final report to Congress. A majority of its Commissioners recommended that computer programs remain copyrightable and proposed an amendment to the recently adopted Copyright Act of 1976 that would include a definition of computer programs[22] and a section limiting the exclusive rights of authors of computer programs where copies must be made to use the program or for archival purposes.[23] Congress amended the Copyright Act in 1980, accepting virtually all of CONTU's recommendations.[24]

The Copyright Act of 1976[25] defines eight categories of materials regarded as "works of authorship," each of which is entitled to copyright protection.[26] Despite the explicit defini-

---

[19] 17 U.S.C. § 102(b) (1992).

[20] Copyright Office Circular 31D (Jan. 1965).

[21] Pub. L. No. 93-573, 88 Stat. 1873 (1974) (codified at 17 U.S.C. § 104 (1988)).

[22] "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." CONTU FINAL REPORT, *supra* note 1, at 30.

[23] *Id.* All computer programs must be copied from permanent memory into working memory before they can be run. If such a use were infringement, computer programs could never legally be used. *See supra* note 13.

[24] Pub. L. No. 96-517, 94 Stat. 3015, 3028 (1980) (codified at 17 U.S.C. §§ 101-117 (1988)). Section 101 of the Copyright Act was amended to include the definition of a computer program, and section 117, concerning the limitation of exclusive rights of computer programs, was replaced. The only modification to CONTU's recommendations for statutory change was to substitute the word "owner" for CONTU's "rightful possessor" in describing who can make copies of a computer program for use or for archival purposes. *See* 17 U.S.C. § 117 (1977 & Supp. I 1989).

[25] 17 U.S.C. § 102(a) (1988 & Supp. IV 1992).

[26] 17 U.S.C. § 102(a) provides:

Works of authorship include the following categories:

(1) literary works;

(2) musical works, including any accompanying words;

(3) dramatic works, including any accompanying music;

(4) pantomimes and choreographic works;

tion of computer programs in the Act and the express limitation on certain rights in computer programs, none of these eight categories explicitly embraces computer software. Congress, however, intended to include computer programs in the category of "literary works."[27] Consequently, literal aspects of computer software, including the sequences of instructions that comprise the source and object codes of a computer program, have been afforded copyright protection without controversy.[28] In addition, flowcharts documenting the general sequence of operations in a computer program also may be afforded copyright protection as long as they are sufficiently original and detailed.[29]

While Congress and the courts have accepted the copyrightability of the literal portions of computer software, controversy has arisen concerning copyright of the "nonliteral" aspects of computer software. Nonliteral elements are not as easily defined or recognized as the literal source and object codes. Nonliteral elements have been said to include the "structure" of computer software as well as certain results of a running program, such as a particular screen display.[30] Copyright

(5) pictorial, graphic, and sculptural works;
(6) motion pictures and other audiovisual works;
(7) sound recordings; and
(8) architectural works.

*Id.*

[27] H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 (1976). "The term 'literary works' . . . includes . . . computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves." *Id.*

[28] CMS Software Design Sys., Inc. v. Info Designs, Inc., 785 F.2d 1246, 1249 (5th Cir. 1986) (source code copyrightable); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983), *cert. denied*, 464 U.S. 1033 (1984) (source and object code copyrightable); Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 454 (N.D. Ga. 1987) (source and object code copyrightable).

[29] A flowchart is a diagram which serves as a roadmap to the programmer. After the general design of the program is agreed upon, it is often reduced to a written flowchart which specifies the sequence and relation of the medium-scale operations of the program. These operations are then fleshed out by the programmer into the hundreds of program instructions required to implement them. "Flow charts, source codes, and object codes are works of authorship in which copyright subsists, provided they are the product of sufficient intellectual labor to surpass the 'insufficient intellectual labor hurdle' . . . ." CONTU FINAL REPORT, *supra* note 1, at 54.

[30] *See* Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1239 (3d Cir.

infringement of nonliteral elements of computer programs is problematic both because of the dispute as to their copyrightability, which turns on whether the elements are ideas or expressions, and the difficulty in ascertaining whether copying actually occurred. In a controversial case, the Third Circuit addressed both of these issues.[31]

Generally, a suit for copyright infringement may be maintained when one of the exclusive rights of the copyright owner, as elaborated in sections 106 through 118 of the Copyright Act, has been violated.[32] To prevail in such a suit, a plaintiff must hold a valid copyright and must prove that the defendant copied the copyrighted work.[33] Most litigation centers on whether copying occurred.

To establish "copying" of the type necessary to prove copyright infringement, the defendant must have actually copied from the plaintiff's copyrighted work, and the copying must go "so far as to constitute improper appropriation."[34] The plaintiff may prove copying through direct evidence (e.g., an admission of copying by the defendant) or circumstantial evidence.[35] Since direct evidence is often unavailable, plaintiffs typically attempt to prove copying by showing (1) that the defendant had access to the copyrighted work and (2) that the defendant's work is "substantially similar" to that of the plaintiff.[36]

Traditionally under the substantial similarity test, the two issues of whether "copying" has occurred and whether such copying is illicit have been distinct and have been governed by different evidentiary rules.[37] When determining whether

1986) (providing copyright protection to a program's structure), *cert. denied*, 479 U.S. 1031 (1987); Stern Elects., Inc. v. Kaufman, 669 F.2d 852, 856 (2d Cir. 1982) (copyrightability of a screen display independent of the underlying program's copyrightability).

[31] *Whelan Assocs.*, 797 F.2d at 1222.

[32] 17 U.S.C. § 501(a) (1992).

[33] *See* Miller v. Universal City Studios, 650 F.2d 1365, 1375 (5th Cir. 1981); Novelty Textile Mills v. Joan Fabrics, 558 F.2d 1090, 1092 (2d Cir. 1977).

[34] Arnstein v. Porter, 154 F.2d 464, 472-73 (2d Cir. 1946) (holding that some copying may not be illicit, and only that which "wrongly appropriate[s] something which belongs to the plaintiff" is actionable), *cert. denied*, 330 U.S. 851 (1947).

[35] *Id.*

[36] *Id.*; *see also* Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1231-32 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987); Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1162 (9th Cir. 1977).

[37] *See, e.g.*, *Arnstein*, 154 F.2d at 468-74.

"copying" has occurred, an analysis of the subject matter of the works is relevant, and both expert and lay testimony are admissible.[38] This is called the "extrinsic" test of substantial similarity. After establishing copying, the analysis then turns to whether an "ordinary lay hearer" believes such copying was illicit; expert testimony is inadmissible on this point.[39] This is the "intrinsic" test of substantial similarity.

When faced with a copyright litigation involving computer software and other complex subject matter, the Third Circuit, in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*,[40] combined the two issues of determining whether copying occurred and whether it was illicit into a single "substantial similarity" inquiry.[41] The court found that "the ordinary observer test . . . which does not permit expert testimony, is of doubtful value in cases involving computer programs on account of the programs' complexity and unfamiliarity to most members of the public."[42] Further, the court felt that the bifurcated approach is ineffective because the finders of fact can hear expert testimony on copying that they must later "forget" in deciding the closely related issue of its illicitness.[43]

In considering the copyrightability of nonliteral elements of computer software, the Third Circuit distinguished ideas from expression in software by holding 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."[44] Thus, every computer program has just one "idea," its primary purpose, and everything surrounding this idea would be copyrightable, as long as it "could be accomplished in a number of different ways."[45] In so narrowly defining a program's idea and in de-

[38] *Id.* at 468-69.
[39] *Id.*
[40] 797 F.2d at 1222.
[41] *Id.* at 1232-33.
[42] *Id.* at 1232.
[43] *Id.* at 1232-33. As the court wrote in *Whelan*, "Especially in complex cases, we doubt that the 'forgetting' can be effective when the expert testimony is essential to even the most fundamental understanding of the objects in question." *Id.* at 1233.
[44] *Id.* at 1236.
[45] *Id.* n.28. The *Whelan* reasoning has been applied specifically to protect nonliteral screen displays. *See* Lotus Dev. Corp. v. Paperback Software Int'l, 740

parting from the traditional lay observer rule of expert testimony, the Third Circuit broke new ground in computer software copyright litigation. Accordingly, this area of copyright law remained controversial when the Second Circuit decided *Computer Associates International v. Altai, Inc.*[46]

## II. *COMPUTER ASSOCIATES INTERNATIONAL V. ALTAI, INC.*

### A. *The Facts*

Computer Associates International, Inc. and Altai, Inc. were both engaged in creating and marketing computer software.[47] One product offered by both companies was a job scheduling program designed to run on IBM System/370 computers.[48] Job schedulers organize a set of tasks for a computer and schedule them to run at times that most efficiently use the computer's resources.[49]

At the time that the two companies were marketing their job scheduling programs, the IBM System/370 (the hardware on which the programs must run) supported three different operating systems.[50] Computer Associates recognized that a single program that could run on any of the three different IBM System/370 operating systems would be far more marketable than a program that was limited to a single operating system.[51] The creation of such a program is problematic, however, because much of the work in writing one version of the program must be triplicated to enable the program to run on

F. Supp. 37 (D. Mass. 1990); Digital Communications Assoc. v. Softklone Dist. Corp., 659 F. Supp. 449 (N.D. Ga. 1987).

[46] 982 F.2d 693 (2d Cir. 1992).

[47] *Id.* at 698.

[48] *Id.* at 698-99. The IBM System/370 is a family of computers designed for use in medium-to-large sized businesses and educational institutions.

[49] *Id.* at 698. For instance, a job scheduler might schedule a time-consuming payroll program to be run at night when a company's employees are not using the computer. Thus, the computer's resources can be dedicated exclusively to employees' work during the day, and the normally unutilized nighttime resources of the computer can be dedicated to the payroll program.

[50] *Id.* Competing operating systems are often sold by different vendors for the same hardware platform, each purporting to offer superior speed, simplicity, etc.

[51] Computer Associates began development of its inter-operating system software, ADAPTER, in 1979. Computer Assocs. Int'l v. Altai, Inc., 775 F. Supp. 544, 552 (E.D.N.Y. 1991).

the two additional operating systems. Further, this triple burden would accrue whenever any other program was to be written for the IBM System/370. Accordingly, any change in one operating system would have to be reflected in the appropriate section of each of the various programs that Computer Associates wrote for the System/370's other operating systems.

The Computer Associates solution was to isolate all of the operating system-dependent computer code in one module, which it called ADAPTER.[52] ADAPTER could be incorporated into any program to allow compatibility with any of the System/370 operating systems and any change in an operating system would require only one change to ADAPTER.[53] Thus, ADAPTER was an ingenious and valuable development which made Computer Associates software more marketable through its compatibility.[54] It was simpler to use because customers no longer needed to purchase a specific operating system or version of the application program, and cheaper to maintain because any changes in an operating system could be made and tested in one small piece of software.[55]

In 1982, Computer Associates began marketing its job scheduler, called CA-SCHEDULER, which incorporated ADAPTER. In the same year, Altai began to market its own job scheduler program called ZEKE.[56] ZEKE, like CA-SCHEDULER, ran on the IBM System/370. Unlike the universal compatibility offered by CA-SCHEDULER, however, ZEKE ran on only one operating system. Responding to customer demand, Altai decided to rewrite ZEKE in 1983 to run on at least one additional operating system.[57]

To accomplish this task, Altai's President, James P. Williams, a former Computer Associates employee, recruited a friend of his, Claude F. Arney, III who still worked at Computer Associates. Williams had not been involved in the development of CA-SCHEDULER or ADAPTER, and did not know that Arney was "intimately familiar with various aspects of

[52] *Computer Associates*, 982 F.2d at 698.
[53] *Id.* at 699.
[54] *Id.*
[55] *Id.*
[56] *Id.*
[57] *Id.*

ADAPTER."[58] In fact, in addition to his work on the ADAPTER project, Arney had been permitted to take copies of ADAPTER's source code home while he was employed at Computer Associates. When he left to work for Altai in 1984, he retained these copies in violation of his employment agreement with Computer Associates.[59]

Arney persuaded Williams that the best solution to providing increased operating system compatibility was to "introduce a 'common system interface' component into ZEKE," which they called OSCAR.[60] Arney failed to mention that this idea originated from his work on ADAPTER, and he never revealed to anyone at Altai that he retained copies of ADAPTER's source code which he used to design and develop OSCAR.[61] After four months of work, OSCAR was complete; Arney had copied approximately thirty percent of its source code from ADAPTER.[62] Altai incorporated a version called OSCAR 3.4 into several of its marketed products, including ZEKE, from 1985 to 1988.

Altai first learned of Arney's possible misappropriations when it received service of process.[63] Arney then confirmed to Williams that the allegations of copying were true, and told Williams which portions of OSCAR 3.4 were copied from ADAPTER. Williams never saw the ADAPTER source code.[64] After seeking legal counsel, Williams decided to rewrite the offending sections of OSCAR 3.4 and exclude Arney entirely from the process. Altai's programmers were given a "description of the ZEKE operating system services" to rewrite the source code.[65] The goal of the project was to save as much of OSCAR 3.4 as legitimately possible while eliminating all portions that had been copied from ADAPTER.[66] After completing the rewrite, the new version, OSCAR 3.5, was sold to

[58] *Id.*
[59] *Id.* at 699-700.
[60] *Id.* at 700.
[61] *Id.*
[62] *Id.* Altai conceded that approximately thirty percent of Oscar was copied directly from the source code of Adapter. Computer Assocs. Int'l v. Altai, Inc., 775 F. Supp. 544, 560 (E.D.N.Y. 1991).
[63] *Computer Assocs.*, 982 F.2d at 700.
[64] *Id.*
[65] *Id.*
[66] *Id.*

Altai's new customers, and old customers were provided with a "free upgrade" of OSCAR 3.4 to OSCAR 3.5.[67]

In July of 1988, Computer Associates brought suit against Altai in the United States District Court for the District of New Jersey, claiming that Altai had misappropriated its trade secrets and infringed its copyright of CA-SCHEDULER in both OSCAR 3.4 and the rewritten OSCAR 3.5.[68] The parties stipulated to transfer the case to the United States District Court for the Eastern District of New York in March 1989.[69]

## B. *The District Court Decision*

The court began its analysis by noting that the focus of the inquiry is whether Altai's OSCAR programs infringed on Computer Associates' ADAPTER program.[70] To prove infringement, a plaintiff must demonstrate ownership of a valid copyright and that the defendant copied the plaintiff's copyrighted work.[71] The court decided that Computer Associates held a valid copyright on ADAPTER even though it was not separately registered. Computer Associates was the undisputed author of ADAPTER and, in fact, held a registered copyright on a version of its job scheduler, CA-SCHEDULER 2.1, which contained ADAPTER. The court found that separate registrations of the sub-modules of a program are not required because "dozens if not hundreds of [such] registrations" would be needed.[72]

Next, the court turned to the plaintiff's burden of proving that the defendant copied the copyrighted work. In the case of OSCAR 3.4, Altai conceded that Arney had copied approxi-

---

[67] *Id.*

[68] The trade secret claim is not addressed in this Comment. However, the original, unreported holding by the Second Circuit that Computer Associates' state law trade secret claims were preempted by section 301 of the Copyright Act was partially reversed in the amended Second Circuit opinion. *See* Computer Assocs. Int'l v. Altai, Inc., No. 762, 91-7893, 1992 WL 139364, at *24-27 (2d Cir. June 22, 1992) (original opinion prior to rehearing).

[69] *Computer Assocs.*, 982 F.2d at 700.

[70] Computer Assocs. Int'l v. Altai, Inc., 775 F. Supp. 544, 555 (E.D.N.Y. 1991).

[71] *Computer Assocs.*, 982 F.2d at 701 (citing Novelty Textile Mills, Inc. v. Joan Fabrics Corp., 558 F.2d 1090, 1092 (2d Cir. 1977)); *see also* Arnstein v. Porter, 154 F.2d 464, 468-74 (2d Cir. 1946), *cert. denied*, 330 U.S. 851 (1947).

[72] *Computer Assocs.*, 775 F. Supp. at 556.

mately thirty percent of the source code directly from ADAPTER and, thus, admitted infringement.[73] There was no direct evidence of copying as to the rewritten OSCAR 3.5, however, so the court addressed whether Computer Associates had proven "'access and substantial similarity between the work . . . [and that the] similarities relate to the copyrightable material.'"[74] The court assumed that Altai, through Arney's copies, did have access to the source code of ADAPTER.

Thus, the question as to whether OSCAR 3.5 infringed upon ADAPTER turned on whether the two programs were substantially similar. The Third Circuit had enunciated its test for substantial similarity in computer programs in *Whelan Associates v. Jaslow Dental Laboratory*.[75] In *Whelan* a dental laboratory program written in a computer language called EDL was essentially rewritten by a competing company in the language BASIC. The court held 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" and, therefore, protectible by copyright.[76] Among the elements of expression that are therefore protectible under the *Whelan* test are the "structure, sequence, and organization" of a computer program.[77] The district court in *Computer Associates* disregarded this test, calling it "inadequate and inaccurate."[78] Bolstered by support from a court-appointed expert and the renowned copyright academic Professor Nimmer, the court found that the *Whelan* test was flawed for two basic reasons. First, it failed to address the notion that a computer program can contain more than one idea. Under *Whelan*, once the single idea that the program embodies is identified, everything contained in the program that is not necessary or incidental to that idea is considered protectible expression.[79] Second, *Whelan* failed to distinguish between

[73] *Id.* at 560.

[74] *Id.* at 557 (quoting Walker v. Time Life Films, Inc., 784 F.2d 44, 48 (2d Cir.), *cert. denied*, 476 U.S. 1159 (1986)).

[75] 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

[76] *Id.* at 1236.

[77] *Id.* at 1248.

[78] *Computer Assocs.* 775 F. Supp. at 559.

[79] *Id.* For example, in *Whelan*, the court determined that the "idea" of the plaintiff's program was "the efficient organization of a dental laboratory." 797 F.2d at 1240. All of "the detailed expression of the . . . program is part of the expres-

the static structure of a program (the text of its source code and sequence of instructions in its object code) and the dynamic structure of a program (the way the program "behaves" when it is running and reacting to input). Because it failed to make this distinction, the *Whelan* test synonymizes the terms "structure," "sequence," and "organization" when in fact these terms are not synonymous.[80]

Further, the *Computer Associates* court noted a possible statutory problem when the dynamic "behavior" of a computer program is recognized. Title 17 section 102(a) provides copyright protection in original works of authorship, including computer programs. Section 102(b), however, provides: "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."[81] The court noted that the behavioral component of a computer program is within the meaning of the words "process," "system," and "method of operation," and that the behavior of a computer program merits no copyright protection at all.[82] The court ultimately did not reach this issue, however, finding that the rights of Computer Associates would be fully preserved by looking at only the static, non-behavioral structure of the ADAPTER program: the text of its source code.[83]

The court followed what it called the "abstractions" test in determining whether the two programs were substantially similar.[84] First proffered by Judge Learned Hand, the abstractions test holds:

sion, not the idea, of that program." *Id.* at 1239. Hence, the entire program was protectible.

[80] *Computer Assocs.*, 775 F. Supp. at 560. Although the Second Circuit recognized that these terms have different meanings, it, too, failed to define them properly. *See infra* part III.A.

[81] 17 U.S.C. § 102(b) (1992).

[82] *Computer Assocs.*, 775 F. Supp. at 560. The court noted the suggestion that computer software might be better protected by patent law rather than copyright law. *Id.*; *see* Randall M. Whitmeyer, Comment, *A Plea For Due Processes: Defining the Proper Scope of Patent Protection for Computer Software*, 85 NW. U. L. REV. 1103, 1123-25 (1991).

[83] *Computer Assocs.*, 775 F. Supp. at 560.

[84] *Id.*

> Upon any work . . . 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 [work] is about and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the [author] could prevent the use of his "ideas" to which, apart from their expression, his property is never extended.[85]

In determining how this test would be applied to computer programs, the court relied heavily on the testimony of the court-appointed expert Dr. Randall Davis of the Massachusetts Institute of Technology's Artificial Intelligence Research Laboratory.[86] The analysis "would progress in order of 'increasing generality' from object code, to source code, to parameter lists, to services required, to general outline."[87]

After stratifying its analysis into these levels, the court, again relying on Dr. Davis's testimony, found that any similarities that existed in the two programs could be explained by: (1) factors external to the program (like the calls which must be made to the operating systems); (2) functional demands (the program's "job"); (3) elements in the public domain; or (4) a structure or method "simple and obvious to anyone exposed to the operation of the program."[88] Hence, OSCAR 3.5 was not substantially similar to ADAPTER and Computer Associates' claim of copyright infringement regarding OSCAR 3.5 failed.[89]

## C. *The Second Circuit Decision*

On appeal Computer Associates contended that the district court's method for determining substantial similarity between ADAPTER and OSCAR 3.5 was in error.[90] The Second Circuit

---

[85] *Id.* (quoting Nichols v. Universal Pictures, 45 F.2d 119, 121 (2d Cir. 1930) (Hand, J.), *cert. denied*, 282 U.S. 902 (1931)).

[86] Dr. Davis composed a written report for the court on basic computer science and varieties of computer software similarity. A revised and expanded version of this report is reprinted in Randall Davis, *The Nature of Software and its Consequences for Establishing and Evaluating Substantial Similarity*, 5 SOFTWARE L.J. 299 (1992).

[87] *Computer Assocs.*, 775 F. Supp. at 560 (quoting *Nichols*, 45 F.2d at 121).

[88] *Id.* at 562.

[89] *Id.*

[90] Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 701 (2d Cir. 1992).

adopted the district court's assumption that Altai had access to Computer Associates' ADAPTER program and, like the district court, narrowed its inquiry to whether the two programs were substantially similar.[91] Unlike the district court, however, the Second Circuit immediately expanded its field of inquiry to include not only the text of the two programs, but the "non-literal components" of the programs as well.[92] It held that "non-literal structures of computer programs are protected by copyright."[93] Finding that Altai's rewrite of OSCAR eliminated the substantial similarity of any "literal" elements of ADAPTER, the court turned to Computer Associates' argument that OSCAR 3.5's *structure* remained substantially similar to that of ADAPTER.

The court reviewed the idea versus expression dichotomy of copyright law, stressing the holding in *Baker v. Selden*[94] that "those aspects of a work, which 'must necessarily be used as incident to' the idea, system or process that the work describes, are also not copyrightable."[95] After rejecting the Third Circuit's test for substantial similarity of computer programs articulated in *Whelan*[96] for much the same reasoning as the district court,[97] the Second Circuit went on to enunciate its own test: "Abstraction-Filtration-Comparison."[98]

The abstraction-filtration-comparison test is quite similar to the methodology applied in the district court's decision. The first step of the tripartite test is to, "in a manner that resembles reverse engineering . . . dissect the allegedly copied program's structure and isolate each level of abstraction contained within it."[99] Next, at each defined level of abstraction, the structural components are examined "to determine whether

[91] *Id.*

[92] *Id.* ("It is of course essential to any protection of literary property . . . that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations.") (quoting Nichols v. Universal Pictures, 45 F.2d 119, 121 (2d Cir. 1930) (Hand, J.), *cert. denied*, 282 U.S. 902 (1931)).

[93] *Computer Assocs.*, 982 F.2d at 702.

[94] 101 U.S. 99 (1879).

[95] *Computer Assocs.*, 982 F.2d at 704 (quoting *Baker*, 101 U.S. at 104).

[96] 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

[97] *Computer Assocs.*, 982 F.2d at 705-06; *see also supra* notes 75-80 and accompanying text.

[98] *Computer Assocs.*, 982 F.2d at 706.

[99] *Id.* at 707.

their particular inclusion at that level was 'idea' or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain."[100] Such components, the court held, are non-protectible.[101] The remaining core of expression left after filtration is subject to copyright protection and is submitted to the third and final step of the analysis: comparison. Here, "the court's substantial similarity inquiry focuses on whether the defendant copied any aspect of this protected expression, as well as an assessment of the copied portion's relative importance with respect to plaintiff's overall program."[102]

Noting that the district court's method of analysis "served as a roadmap" for its own,[103] the Second Circuit affirmed the district court's conclusion that OSCAR 3.5 was not substantially similar to ADAPTER, agreeing that any similar elements were taken from the public domain or dictated by efficiency or external factors.[104]

## III. ANALYSIS

The application of copyright law to computer software has caused a great deal of confusion in courts because most judges do not understand computers and, therefore, are not able to reason confidently how copyright law should apply to computer software. Significantly, decisions often lack a specific, detailed discussion of the computer code at issue in a case. This failing is disturbing for two reasons. First, it may indicate that the fact-finder does not fully comprehend the subject matter in the particular case. Second, and perhaps more importantly, the precedential value of the case as a tool in evidentiary analysis is lost.

To foster an understanding for this technology and its protection, it is imperative that judges and lawmakers alike quash their apprehensions about software and begin to discuss

[100] *Id.* These considerations were espoused by the district court. *See supra* note 88 and accompanying text.

[101] *Computer Assocs.*, 982 F.2d at 710.

[102] *Id.*

[103] *Id.* at 714.

[104] *Id.*

it—in detail. It is troublesome that in many judicial decisions, while the subject matter at issue is computer software, the opinions omit any inclusion of portions of the computer code under scrutiny or even a paraphrased discussion of how the code functions. In the *Computer Associates* decisions there is no specific discussion of any portion of the programs involved.

To protect computer software structure, it is necessary to determine what the structure consists of and what evidence is meaningful in determining whether it has been plagiarized. In this vein, courts must abandon the misleading buzzphrases of "literal" and "nonliteral" expression. Instead, they must be prepared to scrutinize computer code closely to determine both verbatim code similarity and structural similarity. Next, judges will need a well-defined framework for determining substantial similarity. The *Computer Associates* decisions offer some guidance in this regard but not enough. Finally, experts must fill a strictly monitored but important role in aiding the fact-finder in technically detailed analysis.

## A. *Literal vs. Nonliteral: An Unhelpful Distinction*

The pragmatic problem of a court's reliance solely on expert witness testimony, and the difficulties inherent in according copyright protection to nonliteral computer program elements, are illustrated by the Second Circuit's difficulty in defining and addressing nonliteral aspects of computer programs and their relation to a finding of copyright infringement. In *Computer Associates*, the court failed to give a complete, concise definition of "nonliteral components," and in evaluating its expert witness' testimony the court instead focused primarily on literal components of computer programs.

Computer Associates claimed that Altai had infringed the structure of the ADAPTER program. The court found that "a program's structure includes its non-literal components such as general flow charts as well as the more specific organization of inter-modular relationships, parameter lists, and macros."[105] This definition of nonliteral structures embraces the "structure, sequence, and organization" ("SSO") group of elements that many other courts, including the *Whelan* court, have held

[105] *Id.* at 702.

to be protectible by copyright law.[106]

The extent to which these elements of computer program structure are nonliteral is unclear in the Second Circuit opinion. The district court noted that its analysis focused on the programs only as literal text.[107] In its substantial similarity inquiry, the district court relied on Dr. Davis to quantify the relative importance of each of five factors that could indicate similarity between the two programs. Dr. Davis rated the importance of each factor on a scale of 1 to 1,000, and arrived at the following evaluation:

| | |
|---|---|
| •Code | 1000 |
| •Parameter Lists | 100 |
| •Macros | 100 |
| •List of Services | 1 |
| •Organization Chart | Nil[108] |

Dr. Davis and, therefore, the district court found that there was no similarity between OSCAR 3.5 and ADAPTER in the code since OSCAR 3.4 had been rewritten to remove the similar code. Further, most of the parameter lists and macros in ADAPTER were either in the public domain or dictated by the functionality of the program and, thus, were not protectible.[109] Finally, the list of services and organizational chart combined could not alone provide sufficient evidence of substantial similarity to prove infringement.[110] The Second

[106] *See, e.g.*, Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989) (protecting "the non-literal components of a program, including the structure, sequence and organization"); Healthcare Affiliated Servs., Inc. v. Lippany, 701 F. Supp. 1142, 1150 (W.D. Pa. 1988) (noting that copyright protects the "so-called 'non-literal' aspects of [a computer] program, i.e., its structure, sequence and organization."); *see also* Meredith Corp. v., Harper & Row Publishers, Inc., 378 F. Supp. 686, 690 (S.D.N.Y.) (copyright protection extends to the "structure and topical sequence" in textbooks), *aff'd*, 500 F.2d 1221 (2d Cir. 1974).

[107] Computer Assocs. Int'l v. Altai, Inc., 775 F. Supp. 554, 560 (E.D.N.Y. 1991) ("[Computer Associates'] rights . . . are fully protected by viewing the ADAPTER program as text.").

[108] *Id.* at 562.

[109] *Id.* The district court found that "only a few of the lists and macros were similar to protected elements in ADAPTER." *Id.* The Second Circuit held that the district court could find that these few similarities did not constitute infringement "given their relative contribution to the overall program." Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 715 (2d Cir. 1992) (citing the *de minimis* exception in Warner Bros., Inc. v. American Broadcasting Cos. 720 F.2d 231, 242 (2d Cir. 1983)).

[110] *Computer Assocs.*, 775 F. Supp. at 562.

Circuit's review of the district court's analysis focused on this table of relative importance and the finding that the relevant evidence was insufficient to support a claim for infringement.

For its own analysis, the Second Circuit adopted the findings of Dr. Davis and the district court, but in the context of *nonliteral* components of computer software. However, the district court's findings were restricted to the *literal* component context. This semantic extension of the district court's reasoning is misleading because it confuses the distinction between a computer program's code and its structure with a nebulous distinction between so-called literal and nonliteral aspects of software.[111] If courts are not able to discern which components of a computer program should be protected as literal text and which should be protected more abstractly as structure, it will be impossible to apply a meaningful method of comparison to each to determine substantial similarity.

Of the five factors that Dr. Davis evaluated for their importance in assessing substantial similarity, four are defined in the Second Circuit opinion without noting whether each is a literal or nonliteral component of software.[112] The three most important factors, according to Dr. Davis, are the code, the parameter lists and the macros of the program. These are not, however, nonliteral aspects of computer software. All three are literally specified by the code of the program and, in fact, parameter lists and macros are simply segments of code.[113] It is

---

[111] For further criticism of the court's analysis of Dr. Davis' table of importance, see Bruce G. Joseph, *Compilations and Computers, in* ADVANCED SEMINARS ON COPYRIGHT LAW 1992, at 113 (PLI Pat., Copyrights, Trademarks, & Literary Prop. Course Handbook Series No. 336, 1992).

[112] The court defined "code" as the embodiment of the program "in a written language that the computer can read." *Computer Assocs.*, 982 F.2d at 698. A "parameter list" is "'the information sent to and received from a subroutine.'" *Id.* at 697 (quoting Dr. Davis). "A macro is a single instruction that initiates a sequence of operations or module interactions within the program." *Id.* at 698. An "organization chart" or flow chart "map[s] the interactions between modules that achieve the program's end goal." *Id.* at 697.

[113] For example, consider the following line of C code ("C" is a popular programming language):

```
    average(int x, int y)
```

This is the first line of code in a subroutine called "average." A subroutine is a module of code in a program that operates on given inputs to produce a result. Here, "average" would compute the average of two integers ("int") named "x" and "y," which are the inputs or parameters of the subroutine. Everything inside the parentheses (int x, int y) comprises the "parameter list" of the subroutine. For unclear whether the remaining two factors, the list of services and the organization (or "flow") charts, are literal or nonliteral elements. The court treated flow charts as nonliteral elements although they are always depicted in some written format.[114] The court failed to define "list of services."

Thus, according to Dr. Davis's chart of relative importance, the literal factors present in a computer program hold a combined relative weight of 1200, while the possibly nonliteral factors have a relative weight of one. The court's endorsement of this weighting system for determining substantial similarity indicates that the Second Circuit afforded protection to program structure only as it is evidenced by the *literal* aspects of software (which are 1200 times more important than non-literal aspects), that is, the *code* itself. This finding, though only implicit in the court's stated reasoning, transforms the relevant discussion from one concerning literal and nonliteral computer software components to a more meaningful discussion of protecting literal code and literal structure.

The Second Circuit properly recognized that similarities in program structure require a different analysis than do similarities in the verbatim or near-verbatim copying of arbitrary sections of code.[115] Copying of source code can be shown, in the simplest situation, by comparing either source code or object code in a line-by-line fashion to determine whether passages of instructions have been copied either wholesale or in part.[116] As the situation becomes more complex and the literal copying of code sequences are purposefully hidden or dis-

---

purposes of similarity, a macro is equivalent to a subroutine. The primary difference between the two lies in the manner in which the compiler translates the source code into object code.

[114] *Computer Assocs.*, 982 F.2d at 702.

[115] These two distinct types of similarity have been termed "fragmented literal similarity" (verbatim or near-verbatim copying of code) and "comprehensive nonliteral similarity" (structural similarity). 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03[A] (1991); *see also* Eric W. Petraske, *An Infringement Test for Comprehensive Similarity in Software Cases*, COMPUTER LAW., Aug. 1990, at 12.

[116] *See, e.g.*, Customs Service Decision 90-40, 24 Cust. B. & Dec. No. 14, 28 (1990) (using a "side by side comparison" of object code and disassembled source code to reveal verbatim copying of 60-70% of the instructions in an important module); SAS Ins., Inc. v. S&H Computer Sys., Inc., 605 F. Supp. 816, 822 (M.D. Tenn. 1985) (finding "numerous instances of literal [and] near literal . . . copying").

guised by an infringing programmer, the investigation may turn to more indirect indications of copying. For example, replication in the alleged copy of unnecessary instructions or careless errors in the original code may imply copying.[117] Similarly, an unwillingness of the allegedly infringing programmer to take advantage of more powerful hardware resources to which the original program had no access may evidence copying as well.[118]

Infringement in program structure requires a different method of evidentiary analysis than infringement by verbatim copying of code, but the evidence of copying must derive only from the code itself. To say that structure, as applied to computer software, is "nonliteral" is generally an incorrect assertion. The well-worn analogy that protecting the structure of computer programs is like protecting the nonliteral "outline" or "plot" of a play is misguided.[119] Although both computer programs and plays need protection beyond strictly verbatim copying, plays (or poetry, paintings, and other artistic works) convey meanings and feelings and emotions that are intangible and unfixed; that is, the value of the work depends as much on the audience as on the author. For this reason, the search for the boundary at which protectible expression ends and unprotectible idea begins has always been elusive and depends

[117] E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1495-96 (D. Minn. 1985) (three lines of unnecessary instructions appearing in the original and allegedly infringing work imply verbatim copying); *cf.* NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d 1177, 1190 (N.D. Cal. 1989) (copying of a bug and its accompanying "patch" into the microcode of an allegedly infringing microprocessor).

[118] For instance, in *E.F. Johnson Co.*, 623 F. Supp. at 1485, the plaintiff's program used 96% of its Intel microprocessor's speed to make time-critical calculations needed to synchronize its mobile radios with remote transmission facilities. Defendant's program, which needed to make similar calculations, used the same calculation code as the plaintiff's program. The defendant's program, however, ran on a much faster microprocessor and, thus, the defendant's code could have been written to take advantage of the faster processor's speed to increase the overall reliability and efficiency of the defendant's mobile radio. The defendants' failure to make use of this hardware superiority "indicate[d] inferentially that [defendant's] engineers copied the [plaintiff's] code verbatim without considering the utility which the greater speed of the Hitachi microprocessor afforded them." *Id.* at 1494.

[119] Protecting the outline or structure of a play is the subject of Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930), which is often cited as the touchstone for copyright protection of nonliteral expression: "But when the plagiarist does not take out a block in suit, but an abstract of the whole, decision is more troublesome." *Id.* at 121.

strongly on the facts of a given case and an audience's reaction to them. This difficulty arises largely because an artistic author's medium is essentially unstructured and its very purpose is to convey a variety of individual interpretations to a variety of people. Thus, the types of evidence that can be proffered by plaintiffs to show substantial similarity necessarily vary from case to case even when the same general types of works are involved.

Computer programmers, by contrast, do not operate in the intangible, open-ended environments that artistic authors enjoy. Computer software is created in a tightly constrained, highly structured medium: that of a computer language that prescribes many rules and permits no variation or error. Computer languages are not intended to convey a variety of meanings. They can be interpreted in a single way: that by which the computer sequentially executes the instructions in the code. All results are predetermined. Nothing is left to the whimsy of a human audience. Indeed, the Copyright Act acknowledges that a computer program is "a set of statements or instructions to be used . . . in a computer in order to bring about a *certain* result."[120]

Therefore, in evaluating similarities between structures of computer programs, the Copyright Act directs, and programming practices dictate, that the entire evidentiary investigation be limited to the literal instructions that comprise the software. Explorations into vague flow charts created before the code for a program was written, comparison of simple high-level lists of services that two programs provide or acceptance of the copying of instruction manuals as evidence of copying the related code,[121] .do not reveal with any certainty whether two programs are substantially similar. Moreover, because all computer programs are written in the same rigidly defined environment,[122] the types of evidence that can show substantial similarity should remain largely the same from case to

[120] 17 U.S.C. § 101 (1992) (emphasis added).

[121] Such evidence was considered by the court in *E.F. Johnson Co.*, 623 F. Supp. at 1497 ("Verbatim copying of a computer manual is inferential evidence of pirating of the underlying software.") (citing Synercom Technology v. University Computing Co., 462 F. Supp. 1003, 1010 (N.D. Tex. 1978)).

[122] At the very least, those programs that are written in the same language for the same hardware are created in identical environments.

case. Therefore, the goal of creating a clear methodology by which the structures of two programs can be compared should be achievable. Yet neither the district court nor the Second Circuit in *Computer Associates* offered any specific guidelines as to how such a method of comparison should operate.

## B. *The Second Circuit's Tripartite Test for Substantial Similarity*

The Second Circuit in *Computer Associates* endorsed a three part substantial similarity test for program structure based upon traditional copyright doctrines.[123] First, a court must separate the allegedly infringed program into its structural components (the "abstraction" step). Next, uncopyrightable elements, such as ideas, expression necessarily incident to the ideas, and elements in the public domain are filtered from the program (the "filtration" step). Finally, the remaining core of protectible material is compared to the allegedly infringing work to determine whether protected expression has been copied to an extent sufficient to constitute infringement (the "comparison" step).[124]

Although the Second Circuit adopted a more informed view of computer programming practice in its analysis of the idea and expression dichotomy than have other courts, its evidentiary analysis focused primarily on the straightforward step of filtration instead of on the more difficult steps of abstraction and comparison.[125] All three steps are vital to the analysis. Courts, therefore, will need more specific guidance to apply the scantily outlined abstraction and comparison steps in analyzing similarities in program structure.

### 1. Abstraction

The first step in the Second Circuit's substantial similarity analysis, which is also the first step in separating ideas from

[123] "This approach breaks no new ground; rather, it draws on such familiar copyright doctrines as merger, *scenes a faire*, and public domain." Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992).

[124] *Id.*

[125] The court's discussion of the filtration step is three times longer than its discussion of the abstraction and comparison steps combined. *Id.* at 706-11.

expression, is to "dissect the allegedly copied program's structure and isolate each level of abstraction contained within it."[126] Thus, the court recognized that computer programs may contain various levels of structure, ranging from the individual instructions in the code, to low-level subroutines that comprise the program, to larger groups of subroutines that work together as independent modules within the program.[127]

The Second Circuit's acknowledgment that computer programs contain many levels of structure, though intuitive to programmers, is an important departure from a line of cases spawned by the widely criticized opinion in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*[128] In *Whelan* the court concerned itself with identifying the boundary between the protectible expression and the unprotectible idea in a computer program. Relying on *Baker v. Selden*,[129] its conclusion was that "the purpose or function" of a computer program is the idea, and everything in the program that "is not necessary to that purpose or function would be part of the expression of the idea."[130] Thus, *Whelan* held that the only idea that can be embodied in a computer program is that of the simplest, highest level of abstraction: the ultimate purpose of the entire program. Application of this rule is straightforward and predictable, but it extends copyright protection too broadly by ignoring the fact that computer programs can contain many ideas, each embodied in only a portion of the whole program, and

[126] *Id.* at 707.

[127] *See* Steven R. Englund, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 897-98 (1990).

[128] 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987); *accord* Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127 (N.D. Cal. 1986). For a sampling of the criticisms of the *Whelan* analysis, see generally Kretschmer, *supra* note 2. *See also* Cary S. Kappel, *Copyright Protection of SSO: Replete with Internal Deficiencies and Practical Dangers*, 59 FORDHAM L. REV. 699, 707 (1991) (finding *Whelan*'s justifications for its method of protection to be insufficiently grounded in traditional copyright law); Gregory J. Ramos, Lotus v. Paperback: *Confusing the Idea-Expression Distinction and its Application to Computer Software*, 63 U. COLO. L. REV. 267, 274 (1992) (observing the trend of most courts to retreat from the *Whelan* analysis).

[129] 101 U.S. 99 (1879). "Just as *Baker v. Selden* focused on the end sought to be achieved by Selden's book, the line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question." *Whelan*, 797 F.2d at 1236.

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

none of which are subject to copyright protection.

By adapting a multi-level abstractions test, the Second Circuit successfully conformed traditional copyright doctrine to the new technology of computer software. The court drew upon the abstractions test as it was initially set forth by Judge Learned Hand in *Nichols v. Universal Pictures Co.*[131] But instead of striving to determine at which particular level of abstraction expression ends and idea begins, the Second Circuit modified the test to search for idea and expression at *every* level of abstraction. This approach is appropriate in analyzing computer software because unlike conventional artistic and literary works, computer programs possess a well-defined, identifiable hierarchy of organizational levels. In creating a piece of software, a programmer begins by organizing conceptions at a high, functional level. After making conscious design choices, the programmer moves "down" one level in the hierarchy, making more design decisions and organizing progressively smaller submodules to implement the module above. Each level of this hierarchy of organizations is a structure of the program, and each may contain ideas and expressions.

The ability to "reverse engineer"[132] the programming process to recreate the hierarchy of structures is central to the Second Circuit's abstraction test.[133] However, courts will need guidance in carrying out this process. Unfortunately, the Sec-

[131] 45 F.2d 119, 121 (2d Cir. 1930); *see supra* note 85 and accompanying text.

[132] "Reverse engineering" refers to the process of ascertaining the individual components of a system and their interaction by analyzing the finished product. For example, a carpenter could reverse engineer a desk by disassembling the individual parts to discover what parts were used and the pattern in which they were assembled, resulting in a blueprint for the desk. In the software context, this involves disassembling the object code of a program into human-readable source code, which can be further analyzed to determine the patterns in which separable modules of the source code, performing different functions, interact. This assembly of program modules is one level of program structure.

Reverse engineering as an industry practice poses a threat to copyright protection that can be combated with other forms of intellectual property protection. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1521 (9th Cir. 1992) (endorsing reverse engineering as a "fair use" of a copyrighted work under some circumstances); *see also* Philip J. McCabe, *Decision Time on Technology's Frontier*, THE RECORDER, Nov. 4, 1992, at 7 (discussing alternative forms of protection against reverse engineering).

[133] The court likened the abstraction process to "reverse engineering on a theoretical plane." Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 707 (2d Cir. 1992).

ond Circuit was less than generous in giving specific instructions on how this is to be accomplished. The court emphasized the important requirement that "[t]his process begins with the code,"[134] but offered no insight other than an implicit endorsement of Dr. Davis's table of "similarity factors" and their relative importance.

### 2. Filtration

The second step in the Second Circuit's test is to examine the structural components found at each level of abstraction indicated in the first step, and to determine whether they constitute an "idea," were taken from the public domain or were included for reasons of efficiency or factors external to the program, and are thus not protectible.[135] As the court noted, this step defines "the scope of the plaintiff's copyright."[136] The remaining structural components constitute a core of protectible expression[137] and may be submitted to the comparison described in the third step of the test. Unlike the abstraction and comparison steps of the Second Circuit's substantial similarity test, the court gave rather specific instructions concerning the types of elements that must be filtered from each level of abstraction in the effort to separate ideas from protectible expression.

First, structural elements dictated by efficiency must be filtered from each level of structural abstraction.[138] This is perhaps the court's most important instruction because it sensibly corrects any misconceptions concerning the use of the merger doctrine in the computer software context. Under the general doctrine of merger, when there is only one way to express a particular idea, the idea and its expression are considered inseparable. The expression is said to have "merged" with the idea and is thus not protected by copyright.[139] The

---

[134] *Id.*

[135] *Id.*

[136] *Id.* (quoting Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475 (9th Cir. 1992)).

[137] *See* NIMMER & NIMMER, *supra* note 115, at § 13.03[F][5].

[138] *Computer Assocs.*, 982 F.2d at 707-09.

[139] Concrete Machinery Co. v. Classic Lawn Ornaments, Inc., 843 F.2d 600, 606 (1st Cir. 1988).

*Whelan* court had interpreted this to mean that where there are "various" ways of expressing the idea, the particular expression chosen is "not necessary to the purpose" and thus has not merged with the idea.[140] Hence, so long as there are "various" ways of expressing an idea, a *particular* expression will be protectible.[141] *Whelan* qualified this finding by stating that where "there are only a very limited number" of structures available to accomplish the task, there may necessarily be little variation between even independently created works and therefore copyright should not protect such structures.[142]

The Second Circuit took this analysis an important step further by holding that even where there *are* various structures available for accomplishing a certain task, structures which are chosen because they are efficient are not copyrightable. An important goal of both the software industry and copyright law is to distribute better software to users. The Second Circuit recognized that evidence of similar efficient structure therefore may lead to an inference of independent creation as likely as it would to an inference of copying.[143] Hence, the evidence is not probative of copying and should be dismissed or "filtered" from the analysis. The court also noted that two types of efficient structure are unprotected: that which causes a program to run more efficiently within a computer, and that which allows a user to make more efficient use of a computer.[144] This is a substantial and appropriate bar to

[140] Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987). The Second Circuit formulated the general application of merger to computer software as an inquiry into "'whether the use of *this particular set* of modules is necessary efficiently to implement that part of the program's process' being implemented. If the answer is yes, then the expression represented by the programmer's choice of a specific module or group of modules has merged with their underlying idea and is unprotected." *Computer Assocs.*, 982 F.2d at 708 (citing Englund, *supra* note 127, at 902-03).

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

[142] *Id.* at 1243 n.43.

[143] *Computer Assocs.*, 982 F.2d at 708.

[144] Other courts have previously extended the merger doctrine to structures accommodating user efficiency. *See, e.g.*, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 66 (D. Mass. 1990) (the expression of which key on a keyboard should be used to invoke a command system merged with the underlying idea since the key "must be easily accessible"); Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 995-99 (D. Conn. 1989) (expression of a method of allowing a user to navigate between screen displays not protectible since there were a limited number of methods that would "facilitate user comfort").

copyright protection which allows independent creators to feel unencumbered by the law in their efforts to make computer software more efficient and more usable.

The second category of items that must be filtered are elements dictated by external factors.[145] This is simply an appreciation that the computer programming environment is necessarily constrained by the hardware and other programs with which the software will interact, as well as by the demands of the industry or customers that the software will support.[146] Additionally, programmers tend to conform their work to standards set by computer manufacturers and to widely accepted industry programming practices. Because programmers have little or no choice in these areas, structural similarities dictated by these factors are not probative of copying and must also be filtered from the substantial similarity analysis.

The third and simplest category of filtered elements are those in the public domain. By definition, material in the public domain may be taken without remuneration and an author's inclusion of the material in a copyrighted work does not remove it from free, public access.[147] The Second Circuit found no reason to alter this rule for computer software, and noted that programming practices that are "commonplace in the computer software industry" may fall within this category as well as within the category of elements dictated by external factors.[148]

The Second Circuit's application of the filtration step to the facts of *Computer Associates* was minimal. The court merely agreed with the district court judge's finding that almost all similarities between the parameter lists and macros in OSCAR and ADAPTER were due to the "functional demands of the program" (external factors) or to their being in the public do-

[145] *Computer Assocs.*, 982 F.2d at 709.

[146] *Id.*; *see also* Plains Cotton Coop. v. Goodpasture Computer Serv., 807 F.2d 1256, 1262 (5th Cir.) (finding that factors in the cotton market played "a significant role" in determining the structure of cotton marketing software), *cert. denied*, 484 U.S. 821 (1987).

[147] *See* Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54 (2d Cir.) (noting that even where a prior, unprotected work is independently created by another author, the public is still free to copy from the prior work), *cert. denied*, 298 U.S. 669 (1936).

[148] *Computer Assocs.*, 982 F.2d at 710 (citing Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1473 (9th Cir. 1992)).

main.[149] Further, similarities in the two programs' lists of services were also dictated by external factors: the demands of the operating system to which the programs were linked.[150] The district court opinion did not provide any more information concerning the specific application of the filtration process to the evidence in the case than did the Second Circuit opinion. Again, the opportunity to guide evidentiary analysis was lost.

3. Comparison

The final step in the Second Circuit's substantial similarity inquiry involves a comparison between the plaintiff's abstracted, filtered program and the allegedly infringing program.[151] The court's guidance here was slimmest of all. The single focus was on whether the defendant had copied any of the plaintiff's protected expression (*i.e.*, the remaining "core" of material following abstraction and filtration). The court only vaguely noted that an "assessment of the copied portion's relative importance with respect to the plaintiff's overall program" must also be undertaken.[152] But the only elaboration as to the manner in which this assessment relates to the analysis was in a *de minimis* exception by which the court noted that a "few" similarities may not give rise to an infringement given the minor "relative contribution [of the similar entities] to the overall program."[153]

This step of the court's analysis is the most troublesome. The court failed to define what types of evidence are to be compared and, more crucially, how they are to be compared. Are there constructs in source code that, when similar, are more indicative of structural copying than others? What are they? How are they to be weighted? How much variance can exist between two structures that are "similar" in a legal sense? What is the threshold for the *de minimis* exception? These questions were neither asked nor answered in either the district court or Second Circuit *Computer Associates* opinions,

[149] *Computer Assocs.*, 982 F.2d at 714.
[150] *Id.*
[151] *Id.* at 710-11.
[152] *Id.* at 710.
[153] *Id.* at 714-15 (citing Warner Bros. v. American Broadcasting Cos., 720 F.2d 231, 242 (2d Cir. 1983)).

yet they must be asked and answered by each trial judge who faces a software copyright case in the future. A solid body of evidentiary precedent must be created to guide judges through these heavily technical software cases. Of course, such an entity cannot be intelligently created without the consultation of those most familiar with the subject: experts in software design and creation.

## C. *The Proper Use of Expert Testimony*

Under traditional copyright law, the lay observer determined whether the work of one author infringed upon the copyrighted work of another.[154] The lay observer was an appropriate fact-finder when the subject matter of most copyrights consisted of artistic works intended for, and comprehensible by, the general public.

With the advent of copyright in utilitarian works, copyrights were issued for works that were intended not for the general public, but for a specific audience.[155] The public-at-large no longer possessed the specialized knowledge necessary to understand and recognize unauthorized and illicit copying of these works. Consequently, in cases of complex subject matter, some courts have modified the lay observer test to make the finder of fact the intended observer: an individual familiar with the subject matter by being a member of the audience that the author intended to reach.[156] In the context of computer copyright cases, an expert witness either becomes the intended observer or renders significant testimony to guide the ordinary observer.[157]

Computer software is unintelligible to most of the public, and most judges also have no special computer training. Thus, experts in the fields of computer engineering and computer

[154] Arnstein v. Porter, 154 F.2d 464, 468 (2d Cir. 1946), *cert. denied*, 330 U.S. 851 (1947).

[155] Such works intended for a specific audience have included, for example, ledger forms for accountants, or musical works of a specific genre.

[156] *See* Dawson v. Hinshaw Music, Inc., 905 F.2d 731, 737 (4th Cir.), *cert. denied*, 498 U.S. 981 (1990).

[157] *See generally* Robert G. Sugarman, *The Use of Experts and Survey Evidence in Copyright, Trademark and Unfair Competition Litigation*, *in* LITIGATING COPYRIGHT, TRADEMARK, AND UNFAIR COMPETITION CASES 1992, at 301 (PLI Pat., Copyrights, Trademarks & Literary Prop. Course Handbook Series No. 351, 1992).

science are nearly always retained at some point during computer litigation to familiarize the court and, if present, the jury with at least basic concepts. Federal Rule of Evidence 706 permits a court to appoint an expert witness upon its own motion or that of a party.[158] The question is at what point may expert testimony be used, by whom and to what extent.

The district court in *Computer Associates* appointed its own expert witness in addition to considering testimony by the expert witnesses retained by both parties "because of the extensive technical evidence and expert testimony anticipated from both sides."[159] Dr. Randall Davis of the Massachusetts Institute of Technology gave extensive testimony on the creation of computer software, how software controls computer hardware, and his opinions on the similarity of the two programs at issue.[160]

On appeal, Computer Associates claimed that "the district court erred by relying too heavily on the court appointed expert's 'personal opinions on the factual and legal issues before the court.'"[161] In fact, the district court opinion quotes extensively from Dr. Davis's testimony and consistently adopts Dr. Davis's reasoning as its own. In its substantial similarity inquiry alone, the court either quotes or paraphrases Dr. Davis's opinions at least sixteen times. The court praised Dr. Davis's "convincingly demonstrated" arguments on the flaws in the *Whelan* case,[162] allowed Dr. Davis to "invite[] further evidence from the parties' experts"[163] and frequently drew conclusions of fact from Dr. Davis's opinions without further inquiry or analysis.[164] These actions suggest that Dr. Davis did

[158] Rule 706(a) provides:

> Appointment.—The court may on its own motion or on the motion of any party enter an order to show cause why expert witnesses should not be appointed, and may request the parties to submit nominations. The court may appoint any expert witnesses agreed upon by the parties, and may appoint expert witnesses of its own selection . . . .

FED. R. EVID. 706(a).

[159] Computer Assocs. Int'l v. Altai, Inc., 775 F. Supp. 544, 549 (E.D.N.Y. 1991).

[160] *Id.* at 549-51, 558-62.

[161] Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992).

[162] *Computer Assocs.*, 775 F. Supp. at 559.

[163] *Id.* at 561-62.

[164] For example, without further elaboration, the court found that "the rewriting process [of OSCAR 3.4 to OSCAR 3.5] was, *as Dr. Davis testified*, a reasonable method." *Id.* at 562 (emphasis added).

not merely assist the trier of fact; he, in fact, became the trier of fact.[165]

The Second Circuit reviewed the evolving law on the subject of expert testimony in cases of copyrights on complex subject matter and held that the extent to which expert testimony may be used by the fact-finder in a case involving computer programs is wholly within the discretion of the district court.[166] Noting that "Dr. Davis' opinion was instrumental in dismantling the intricacies of computer science," the court concluded that the district court "remained, in the final analysis, the trier of fact" despite its heavy reliance on the expert.[167] With all of Dr. Davis's expert testimony thus admissible, the Second Circuit agreed with the district court that the few elements in the two programs that might be construed as being copied "did not warrant a finding of infringement given their relative contribution to the overall program."[168] The estimation of these elements' relative contribution came from a quantitative evaluation by the expert, Dr. Davis.[169] Thus, the Second Circuit endorsed the use of expert testimony in both the extrinsic and intrinsic prongs of the substantial similarity inquiry.[170]

Combining the traditional two-prong substantial similarity inquiry into a single test uniformly admitting expert testimony is a growing trend in computer copyright cases.[171] Some courts prefer to maintain the theoretical distinction between the two tests, but arrive at the same result. In this approach,

---

[165] Victoria Slind-Flor, *Tackling High Tech,* NAT'L. L.J., Oct. 19, 1992, at 1 ("Professor Davis wrote a 60-page paper on software technology that so impressed Judge Pratt, and the Second Circuit panel that handled the case's appeal, that rulings from both courts quoted the report extensively.").

[166] Computer Assocs. Intn'l v. Altai, Inc., 982 F.2d 693, 712-14 (2d Cir. 1992). Curiously, the court quoted Whelan Assoc. v. Jaslow Dental Lab. Inc., 797 F.2d 1222, 1233 (3d Cir. 1986): "an integrated test involving expert testimony and analytic dissection may well be the wave of the future in this area."

[167] *Computer Assocs.*, 982 F.2d at 714.

[168] *Id.* at 714-15; *see also Computer Assocs.*, 775 F. Supp. at 562.

[169] *Id.* at 562.

[170] *See supra* notes 37-39 and accompanying text.

[171] *See* Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1474 n.3 (9th Cir. 1992) (endorsing a test "in which lay and expert testimony are uniformly admissible" in cases of computer copyright); *Whelan*, 797 F.2d at 1233 ("We . . . adopt a single substantial similarity inquiry according to which both lay and expert testimony would be admissible."); *supra* notes 40-43 and accompanying text.

the "lay observer" fact-finder in the intrinsic prong is instead replaced by the "ordinary reasonable person" who, in computer cases, is a computer programmer. Consequently, the court retains an expert computer programmer to give testimony about the intrinsic prong of the test as well.[172]

Replacing the lay observer with experts in all phases of the substantial similarity inquiry is procedurally necessary, but, at the same time, inherently dangerous. Without the knowledge and experience of experts, the lay fact-finder would drown helplessly in the apparent gibberish of symbols and phrases that comprises computer software. Few computer experts, however—even those appointed by the court with the consent of all parties, like Dr. Davis—are wholly unbiased as to what degree software merits protection by copyright law or other legal doctrines. In fact, Dr. Davis's report for the *Computer Associates* court suggests that he believed that the trend at the time of the litigation to grant liberal copyright protection to software needed to be reexamined.[173] Whether Dr. Davis's opinion is meritorious is only secondarily important. The primary concern is that the political opinion of one expert, which influenced a report relied upon heavily by a court requesting it, may have determined the outcome of this case.

The potential for this situation to arise in any software case (assuming that all such cases will necessitate the use of experts) places a judge in the unhappy position of both playing fact-finder and policing the experts. A judge must and should seek the opinions of several experts to determine what factors are relevant in the substantial similarity analysis and in what quanta they are present in the case at hand. At the same time, a judge must at all times be leery of three activities that could

[172] *See* Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 844 (Fed. Cir. 1992).

[173] Davis, *supra* note 86, at 314 (criticizing the decision in *Whelan*, the then-leading case concerning copyright protection for nonliteral software elements); *see also* COMPUTER SCIENCE & TELECOMMUNICATIONS BOARD, NATIONAL RESEARCH COUNCIL, INTELLECTUAL PROPERTY ISSUES IN SOFTWARE 12-13 (National Academy Press, 1991) (quoting Professor Randall Davis: "We need to think again and we need to be willing to question some of the most fundamental assumptions of intellectual property law."). For a general criticism of the court's use of expert testimony in the *Computer Associates* cases, see Anthony L. Clapes & Jennifer M. Daniels, *Revenge of the Luddites: A Closer Look at Computer Associates v. Altai*, COMPUTER LAW., Nov. 1992, at 11.

render a judgment inequitable: (1) as in all cases, experts must not be biased by an alliance to one of the parties to the suit; (2) especially in the multi-billion dollar international software industry, experts must not be biased by their own political agendas or those of organizations to which they belong; and (3) a judge must, in the end, remain the sole trier of fact. The *Computer Associates* decision may, as some have suggested, have suffered from a district court judge who, perhaps without any alternative, ceded his fact-finding authority to a single expert with preconceived opinions as to software copyright law policy and its application.[174] Without conscientious evidentiary investigation by judges and thorough evaluation of the opinions, motivations and credentials of several experts, uniform and equitable decisions in software copyright cases will remain woefully sporadic.

## D. *The Need for Further Guidance*

Because the Second Circuit's substantial similarity test is so unfamiliar and counter-intuitive,, judges require access to citable precedent that contains relatively detailed discussion of the test's application to specifically illustrated programs. Such a body of law will familiarize judges (as well as lawyers) with computer software and will help develop unified standards of applying the three-part test. Most people, however, are unable to understand source code, and the background necessary to acquire such an understanding is obviously beyond the scope of a judicial opinion. Thus, an analysis of the programs at issue in each case must be presented in an intelligible format.

One possible method for presenting software in a useful manner is in a text-diagram appendix which can be referred to in the text of an opinion as necessary. In the first section, a relevant sample of complimentary code from the plaintiff's and defendant's software should be presented in full. This would allow a side-by-side comparison to determine initially whether identical similarity is present. In the second section, a diagram should be depicted showing the relation of various functional components of the plaintiff's program to one another and to the higher-level purposes of the program. Descriptions of such

[174] Clapes & Daniels, *supra* note 173, at 11.

components may be in plain English and, when properly arranged, pictorially represent the result of the court's "abstraction" step. The third section should show both the source code and the diagram of the plaintiff's program after deleting the elements eliminated by the "filtration" step. Finally, the source code and a component-relation diagram (similar to that in the second section) of the defendant's program must be presented to aid in appreciating the "comparison" step of the court's analysis.

The usefulness of the information provided in such an appendix cannot be gainsaid. Although abstract in form, functional diagrams of the software at issue will enable many more readers to comprehend and analyze the opinion and apply its reasoning to a later case. Further, by juxtaposing diagrams and the related source code, readers will be better able to appreciate the difficulties and constraints that programmers face in implementing particular functions. Moreover, the infusion of actual source code into judicial opinions will at last yield clues to the quantum and quality of evidence necessary to sustain a finding of substantial similarity. Finally, the guidance provided by the presentation of such evidence in judicial opinions will alleviate much of the intimidation caused by the subject matter of these cases.

There is, of course, one countervailing consideration relating to the presentation of the source code in a case: it may also be protected as a trade secret. A party who has developed software in secrecy from competitors may be averse to divulging it in a legal opinion. But computer software evolves rapidly. Programs that may be valuable to a software company today might become obsolete to its users and, thus, valueless to the company tomorrow. In the protracted context of litigation, a party to a copyright action may well find that its software has become useless or obvious by the time a decision in the case is ready for publication. Hence, such a party may be willing to release a relevant portion of the source code for publication in an opinion. Given that predictable, consistent copyright laws are in the interest of all authors of software, the incentive to allow such publication is significant.

## CONCLUSION

With a national industry at stake, courts, or perhaps Congress, must quickly come to grips with the difficulties inherent in the protection of computer intellectual property. It is no longer sufficient to hide unfamiliarity with the complex subject matter of computer software in broad theoretical diatribes that somehow attempt to strain a fluid, dynamic technology into the rigorous traditional definitions of copyright doctrine.

The Second Circuit has made a significant theoretical contribution to copyright law with its tripartite test for substantial similarity in computer programs. Yet, the Second Circuit has done nothing to clarify the application of the test, or to show the court's own ability to apply the test to the facts of the case that brought it about. Further, the Second Circuit has endorsed blind reliance on a single expert for both factual analysis doctrinal creation. To avoid these pitfalls to an otherwise useful analytic framework, district court judges must now strive to become comfortable talking and writing about software and must implement procedures to ensure that expert testimony is unbiased and that its evaluation is competent.

*Andrew G. Isztwan*