Washington Law Review

Volume 68 | Number 2

4-1-1993

A Square Peg in a Round Hole: The Proper Substantial Similarity Test for Nonliteral Aspects of Computer Programs

David A. Loew

A SQUARE PEG IN A ROUND HOLE: THE PROPER SUBSTANTIAL SIMILARITY TEST FOR NONLITERAL ASPECTS OF COMPUTER PROGRAMS

David A. Lowe

Abstract: Since the Third Circuit's decision in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.* expanded copyright protection to include the nonliteral aspects of computer programs, courts have struggled to find a way to properly determine substantial similarity between programs, a necessary element of copyright infringement. In the Third Circuit, courts dissect competing programs and compare them in a one-step procedure. The Ninth Circuit uses a two-part process to objectively, and then subjectively, compare program elements. In *Computer Associates International, Inc. v. Altai, Inc.* the Second Circuit recommended a three-part substantial similarity test to filter out unprotectable elements and compare the remaining core expression. This Comment argues that existing tests are inadequate because they fail to consider programs as a whole and use inappropriate analytical techniques resulting in the loss of protection for nonliteral elements. This Comment proposes the use of a two-tiered substantial similarity test that considers the computer program as an integrated whole and as separate components. This approach would enable courts to fully protect programmers' rights in the original nonliteral aspects of their computer programs.

Compuhypo, Inc., a small computer software company, created and began marketing a new educational word processing program for children designed to teach the alphabet while expanding the child's vocabulary. Called "KidPrint," the program performs four operations common to most word processors: data entry, data display, spell-checking, and printing. Beyond simply displaying typed text, however, KidPrint displays colorful animal pictures on the screen and emits related animal sounds corresponding to various alphabetical characters. Six months after KidPrint hit the stores, Microhypo, Inc., another small computer software company, released a new educational word processing program, "KiddieType," which performed the same operations as KidPrint. Unbeknownst to Compuhypo, Microhypo obtained copies of KidPrint's computer code and drew extensively from it when creating the logic and structure of KiddieType. Compuhypo brought suit against Microhypo, alleging that Microhypo copied significant nonliteral aspects of KidPrint in its KiddieType program. While the literal computer code and each of KiddieType's discrete components are distinguishable from KidPrint's, KiddieType's nonliteral aspects—based on the original selection, coordination, and arrangement of the program and on structure within the components—are virtually identical to KidPrint's. Compuhypo's creative development lies primarily in these nonliteral program aspects.

An owner may prove copying of a computer program by showing that the defendant had access to the copyrighted program and that substantial similarity exists between the two programs. When struggling to define substantial similarity between the nonliteral aspects of computer programs, courts have created various tests. These tests incorporate specific ways to breakdown and compare computer programs.

Although recent substantial similarity tests demonstrate an increased judicial awareness of the technology involved in computer programming, this Comment argues that existing tests are inadequate and undermine copyright protection for the nonliteral aspects of computer programs. Part I discusses copyright protection for computer programs. Part II examines the existing substantial similarity tests courts use to determine whether a computer program has been copied. Part III criticizes those tests in the context of the hypothetical Kid-Print program. Part IV proposes a two-tiered test that addresses those criticisms and aids the courts in determining, comparing, and protecting a programmer's original work.

I. COPYRIGHT PROTECTION FOR COMPUTER PROGRAMS

When courts apply copyright principles to determine infringement between computer programs, they employ the technology of the programs as well as the intricacies of copyright law. To understand the resulting judicial analyses and copyright infringement tests, a reader must have a basic understanding of computer program structure, concepts of copyright law, and the complications that arise when courts apply copyright principles to computer programs.

A. The Structure of Computer Programs

A computer program may be a word processing program, spreadsheet program, database, or other program that the user loads into a computer system[1] and runs by typing the program's name at the command prompt or "clicking" on its program icon with a mouse. To the

1. A computer system consists of both hardware and software. Hardware refers to the physical components of the computer and includes a central processing unit (CPU)—electronic circuits that control the computer and perform its basic calculations and operations—together with memory and input/output devices. Software refers to the instructions that direct a computer to perform operations. Software is composed of one or more computer programs integrated to perform a given task or solve a particular problem. *See* NELL DALE & SUSAN C. LILLY, PASCAL PLUS DATA STRUCTURES 2–3 (2d ed. 1988); D. M. ETTER, STRUCTURED FORTRAN 77 FOR ENGINEERS AND SCIENTISTS 2–3 (2d ed. 1987).

user, a program appears as visual images and audio sounds that provide an interface with which the user accomplishes a given function or achieves a desired result. These audio and visual elements generated by the computer program are referred to as the program's audiovisual aspects.[2] A computer program, however, is much more than its audiovisual representations. A computer program is the complex combination of statements, instructions, and commands created by its programmer.[3]

One must understand how a program is created to understand computer program structure. Typically, a computer programmer creates a program by following a planned sequence of steps.[4] First, the programmer must identify the problem that the computer program attempts to solve.[5] The programmer devises a general strategy for solving the problem by dividing the complex problem into smaller, more manageable subproblems.[6] The programmer solves each of these subproblems using program components called modules.[7] Next, the programmer breaks down these larger modules into smaller modules or subprograms to perform individual tasks.[8] These subprograms interact with each other, with various file structures, and within the larger program modules to solve the problem. The interrelationship among the program's "components"—the modules, subprograms, and file structures—defines the program's "structure."[9] Selecting and arranging program components and creating this interrelationship is the most difficult step in the programming process.[10] It is also the most creative step because here the programmer refines the program, ensuring that data passes in a logical and efficient manner to and from the computer user and between the individual program components.[11]

2. *See* Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 703 (2d Cir. 1992).

3. *See* ROGER S. PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH 10–14 (1982).

4. DALE & LILLY, *supra* note 1, at 11. This methodology is often referred to as the "top-down" approach to computer program design. The programmer defers the specific details of the program as long as possible, dividing its complex requirements into more manageable elements. *Id.; see also* CARLO GHEZZI ET AL., FUNDAMENTALS OF SOFTWARE ENGINEERING 115 (1991).

5. DALE & LILLY, *supra* note 1, at 2, 11.

6. *Id.*

7. *See* PRESSMAN, *supra* note 3, at 148–49.

8. DALE & LILLY, *supra* note 1, at 12.

9. PRESSMAN, *supra* note 3, at 129.

10. *Id.* at 143 (describing this step as the "technical kernel of software [program] engineering").

11. Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc. 797 F.2d 1222, 1230–31, 1237 (3d Cir. 1986) (concluding that the arrangement of modules and subprograms, along with the development of program structure, is the most creative aspect of program creation), *cert. denied,* 479 U.S. 1031 (1987); *see also* DALE & LILLY, *supra* note 1, at 3 (explaining that despite the use

Finally, the programmer converts the developed program into a set of statements or instructions using a written computer language.[12] The programmer writes or "codes" these statements or instructions—commonly referred to as source code[13]—according to the syntax of the chosen computer language.[14] Once the programmer completes the coding of the program, a compiler program translates the source code into computer executable object code.[15]

For the purposes of copyright law, a computer program consists of both literal and nonliteral elements. A program's literal elements are its source and object code.[16] The nonliteral elements of computer programs, however, are not as readily defined.[17] When defining nonliteral elements, courts have included program file structures, screen outputs, and program subroutine interaction;[18] the overall "look and feel" of the program;[19] and program design flow charts, the organization of inter-modular relationships, parameter lists,[20] macros,[21] and a program's list of services.[22] The creation of the nonliteral elements takes place primarily in the program design stage of development in which

---

of common programming knowledge and techniques, program structure is the result of a high degree of creative programming); PRESSMAN, *supra* note 3, at 85 (suggesting that much of the programmer's creative effort lies in designing the program's structure).

12. *See* DALE & LILLY, *supra* note 1, at 12. Modern programmers typically write computer programs in high-level programming languages, examples of which include FORTRAN, COBOL, Pascal, and C. *See* GHEZZI ET AL., *supra* note 4, at 480.

13. Source code refers to a computer program written in a programming language that uses complex symbolic names and rules of syntax. *See* ETTER, *supra* note 1, at 4–7.

14. DALE & LILLY, *supra* note 1, at 1.

15. Object code, written in a binary language comprised of ones and zeros, is source code in machine readable form. *See* ETTER, *supra* note 1, at 4–7.

16. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992); Whelan Assoc., Inc. v. Jaslow Dental Lab., Inc. 797 F.2d 1222, 1234 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

17. *See, e.g., Whelan,* 797 F.2d at 1233–34 (defining nonliteral program elements in a somewhat circular fashion as the tangible or fixed elements of a program's structure that exist beyond the program's literal source and object code but nevertheless relate to the programmer's original program).

18. *Id.* at 1242–46.

19. Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989) (suggesting that the nonliteral elements of program structure include the design of the video screen and the manner in which the computer presents information to the user).

20. A parameter list is the form in which information critical to the function of each module is passed, upon request, among modules in a program. *Altai,* 982 F.2d at 697–98.

21. A macro is a user programmable keystroke sequence used to represent the longer and more complex set of keystrokes normally required to initiate a program response. *Id.* at 698.

22. *See id.* at 697–98, 702.

the programmer tailors the program to the demands of various practical and commercial restraints.[23]

B. Principles of Modern Copyright Protection

The Copyright Act of 1976 explicitly protects authors' original works.[24] To obtain this copyright protection, an author's work must meet two fundamental criteria: the work must be original[25] and fixed[26] in a tangible form.[27] Copyright law only protects an author's original expression and not the ideas or processes used to create the expression.[28] Furthermore, the Copyright Act gives the author the exclusive right to reproduce and distribute the work and to create other works based on the original copyrighted work.[29]

The Copyright Act includes both literary works[30] and audiovisual works[31] as copyrightable works.[32] It also protects compilations[33] and

23. These immediate considerations include efficiency concerns (making the program run faster) and hardware constraints (making the program work with the computer system). Programmers must also design their programs to meet future concerns such as maintainability (modifications made to the program after its initial release); portability (the program's ability to run using different hardware and operating systems); and interoperability (the program's ability to coexist and cooperate with other systems). *See generally* GHEZZI ET AL., *supra* note 4, at 19–36.

24. 17 U.S.C.A. § 102(a) (West Supp. 1992) ("Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device.").

25. A work is original if it is independently created by the author and possesses at least some minimal degree of creativity. Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc., 111 S. Ct. 1282, 1287 (1991).

26. A work is fixed when "its embodiment . . . by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration." 17 U.S.C.A. § 101 (West 1977).

27. *Id.* § 102(a) (West Supp. 1992); *see also Feist,* 111 S. Ct. at 1292–93.

28. 17 U.S.C.A. § 102(b) (West Supp. 1992) provides that "[i]n 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."

29. *Id.* § 106 (West 1977 & Supp. 1992).

30. " 'Literary works' are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects such as books, periodicals, manuscripts, phono-records, film, tapes, disks, or cards, in which they are embodied." *Id.* § 101 (West 1977).

31. " 'Audiovisual works' are works that consist of a series of related images which are intrinsically intended to be shown by the use of machines or devices such as . . . electronic equipment, together with accompanying sounds, if any, regardless of the nature of the material objects . . . in which the works are embodied." *Id.*

32. *Id.* § 102(a) (West Supp. 1992).

33. "A 'compilation' is a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship." *Id.* § 101 (West 1977).

derivative works[34] as copyrightable works.[35] The Act, however, protects only the author's original contributions and not the pre-existing material employed in the work.[36] Copyright protection for compilations and derivative works therefore extends only to the original coordination and arrangement of the work's materials.[37]

The Copyright Act also provides a defendant with certain defenses to a copyright infringement claim.[38] A defendant may escape liability by showing that the allegedly infringing work is original or that the similarities found between the competing works are based on unprotectable elements of the work.[39] In computer program infringement suits, courts have generally recognized three categories of unprotectable program elements: elements dictated by efficiency;[40] elements dictated by external factors;[41] and elements taken from the public domain.[42]

C. *Protecting Computer Programs Under the Copyright Laws*

In the 1976 Copyright Act, Congress failed to explicitly provide for the protection of computer programs. In 1980, upon the recommen-

34. "A 'derivative work' is a work based upon one or more preexisting works . . . which, as a whole, represent[s] an original work of authorship." *Id.*

35. *Id.* § 103(a).

36. *Id.* § 103(b) ("The copyright in a compilation or derivative work extends only to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work, and does not imply any exclusive right in the preexisting material.").

37. *Id.; see also* Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc. 111 S. Ct. 1282, 1289–94 (1991) (holding that the copyright for a factual compilation is limited to the particular selection or arrangement of elements).

38. 17 U.S.C.A. §§ 106–118 (West 1977 & Supp. 1992) (detailing the scope and limitations of an owner's copyright protection).

39. *See Feist,* 111 S. Ct. at 1290–92.

40. Computer program elements are dictated by efficiency when there is essentially only one way to express the idea underlying a specific programming task. The element's expression is said to merge with the task's overall purpose or function. Often referred to as the merger doctrine, the resulting program element expression is considered unprotectable. *See* Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707–09 (2d Cir. 1992); Data East U.S.A., Inc. v. Epyx, Inc., 862 F.2d 204, 208 (9th Cir. 1988).

41. Program elements are dictated by external factors when a programmer's freedom of design choice is limited by extrinsic considerations such as computer hardware compatibility requirements. This is also known as the *scenes a faire* doctrine. *See Altai,* 982 F.2d at 709–10; *Data East,* 862 F.2d at 208; Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987); *see also supra* note 23 and accompanying text.

42. Program elements are taken from the public domain when they consist of material, either original or previously copyrighted, which is freely circulated to the public through program exchanges or similar information-disbursement functions. Public domain material cannot be appropriated by a single author despite its inclusion in a copyrighted work. *See Altai,* 982 F.2d at 710; *Whelan,* 797 F.2d at 1232 n.23, 1248 n.47.

dation of CONTU,[43] Congress expressly included computer programs within the copyright statute.[44] Instead of creating a separate category, however, Congress classified computer programs as "literary works" already protected under copyright law.[45] Congress also provided copyright protection for computer screen displays generated by the literal source code as entirely distinct works of authorship—audiovisual works.[46] Under existing law, copyright protects the literal expression of computer programs[47] such as the program's source code and corresponding object code.[48] Copyright also protects the nonliteral expression of computer programs.[49] Courts disagree, however, on the nonliteral program elements eligible for copyright protection.[50]

Furthermore, copyright law only protects an author's original expression and not the ideas or processes used to create the expression.[51] This critical distinction becomes hazy when courts apply it to computer programs. Courts may view computer program elements as abstract ideas about how to accomplish specific tasks or as computational processes describing how to perform those tasks; copyright law denies protection in either instance, although perhaps for different rea-

43. CONTU is short for the National Commission on New Technological Uses of Copyrighted Works. FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS 1 (1978) [hereinafter CONTU REPORT]. Congress created the Commission in 1974 in response to anticipated problems in the application of its pending copyright bill to computers and other new technologies. *Id. See generally* MARSHALL LEAFFER, UNDERSTANDING COPYRIGHT LAW § 3.4[A] (1989); 2 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 8.08 (1992). CONTU's final report recommended express statutory copyright protection for computer programs to the extent that they embody a programmer's original creation. CONTU REPORT, *supra,* at 1.

44. 17 U.S.C.A. § 101 (West Supp. 1992).

45. Congress defined a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result," *id.,* thereby including programs within the definition of "literary works." *See* H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667 [hereinafter HOUSE REPORT] (stating that the Copyright Act's definition of "literary works" included "computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves"). *See generally* 1 PAUL GOLDSTEIN, COPYRIGHT § 2.15.2 (1989) (describing congressional intent and judicial decisions recognizing the copyrightability of computer programs as "literary works").

46. 17 U.S.C.A. §§ 101, 102(a)(6) (West 1977 & Supp. 1992).

47. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992); Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1233 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983), *cert. dismissed,* 464 U.S. 1033 (1984).

48. *See supra* notes 13–15 and accompanying text.

49. *See Altai,* 982 F.2d at 702–03; Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989); *Whelan,* 797 F.2d at 1248.

50. *See supra* notes 17–22 and accompanying text.

51. *See supra* note 28 and accompanying text.

sons.[52] Courts may also view computer program elements as original expression eligible for copyright protection.[53] Because computer programs fall between classical divisions of unprotectable and protectable works, courts have been forced to analyze them as legal hybrids.[54] As a result, courts have created inconsistent standards for distinguishing between a program's uncopyrightable idea or process and its copyrightable original expression.[55]

II. THE SUBSTANTIAL SIMILARITY TEST FOR COMPUTER PROGRAMS

A copyright owner may bring a copyright infringement suit for a violation of the Copyright Act.[56] To prevail, the owner must show two specific elements: (1) ownership of a valid copyright,[57] and (2) copying of original elements of the copyrighted work.[58] The owner may prove copying of the work's expression by showing the defendant's access to the copyrighted work[59] and substantial similarity between the defendant's work and the owner's copyrighted work.[60] Due to the problem of distinguishing an idea or process from an expression, courts have difficulty determining what elements of computer programs are copyrightable and, in turn, whether substantial similarity exists between elements of competing computer programs.[61]

52. Ideas are not protected per se, *see* Baker v. Seldon, 101 U.S. 99 (1879), while processes are protected, if at all, through patent law. *See* 35 U.S.C.A. § 101 (West 1984). *See generally* DONALD S. CHISUM & MICHAEL A. JACOBS, UNDERSTANDING INTELLECTUAL PROPERTY LAW § 2C[1] (1992) (discussing processes as patentable subject matter).

53. *See supra* notes 24, 45 and accompanying text.

54. *See, e.g.*, Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992) (describing "the hybrid nature of a computer program, which, while it is literary expression, is also a highly functional, utilitarian component in the larger process of computing"); *see also* J.H. Reichman, *Computer Programs as Applied Scientific Know-how: Implications of Copyright Protection for Commercialized University Research*, 42 VAND. L. REV. 639, 656–62 (1989) (discussing how computer programs are legal hybrids falling between the classical forms of protection for industrial and artistic property).

55. *See Altai*, 982 F.2d at 704–05; Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1234–38 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

56. 17 U.S.C.A. § 501(b) (West Supp. 1992).

57. For a complete discussion of the ownership element, see 3 NIMMER & NIMMER, *supra* note 43, § 13.01[A]; 2 GOLDSTEIN, *supra* note 45, § 14.3.1.

58. Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc., 111 S. Ct. 1282, 1296 (1991); *see also* Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 548 (1985).

59. For a detailed discussion of access, see 3 NIMMER & NIMMER, *supra* note 43, § 13.03[D]; 2 GOLDSTEIN, *supra* note 45, § 7.2.1.

60. *See* Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 701 (2d Cir. 1992); Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1472 (9th Cir.), *cert. denied*, 113 S. Ct. 198 (1992).

61. *See, e.g., Altai*, 982 F.2d at 703–05; Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 54 (D. Mass. 1990).

Three circuits have articulated three different standards for determining substantial similarity between computer programs: the Third Circuit's one-step integrated test; the Ninth Circuit's bifurcated test; and the Second Circuit's three-part test.

A. Whelan's *One-Step Integrated Test*

In *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,[62] the Third Circuit recognized that copyright protection extends beyond a computer program's literal code to its nonliteral "structure, sequence, and organization."[63] The court adopted a one-step test to determine if similarity exists between nonliteral aspects of computer programs.[64] Under this test, a court determines the program's ultimate purpose or function, the equivalent of the program's idea.[65] The court will not protect the program's idea, but it will protect all remaining literal and nonliteral elements as copyrightable expression.[66] The court then compares the two competing programs to determine whether substantial similarity exists.[67] In *Whelan,* the court found that the overall idea of the copyrighted program—the efficient management of a dental laboratory—was not protected.[68] Because this idea could be accomplished in a variety of different ways, however, the court concluded that the program's structure was protectable expression rather than unprotectable idea,[69] and found substantial similarity based on three distinct nonliteral program elements: the file structures, screen outputs, and the structure of five specific subroutines of the programs.[70]

B. The Ninth Circuit's Bifurcated Test

The Ninth Circuit has rejected *Whelan's* one-step integrated test in favor of a two-part substantial similarity analysis generally referred to

62. 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

63. *Id.* at 1248. Analogizing the copyrightability of nonliteral aspects of compilations and derivative works to the nonliteral aspects of computer program structure, the court held that as long as an alternate means of expressing a program idea exists, the structure, sequence, and organization of a program is not essential to the program's task and is therefore protectable expression. *Id.* at 1238.

64. *Id.* at 1233.

65. *Id.* at 1236 ("[T]he 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.").

66. *See id.*

67. *See id.* at 1232–33.

68. *See id.* at 1238–40.

69. *Id.* at 1236 n.28.

70. *Id.* at 1242–46.

as the "extrinsic-intrinsic" test.[71] The "extrinsic" part of this test requires an objective analysis of expression.[72] To determine whether similarity exists between computer programs, the trier of fact breaks down or analytically dissects the two competing programs into standard program elements such as screens, menus, and keystrokes and compares corresponding elements.[73] If similarity exists between these program elements, the trier of fact performs the "intrinsic" part of the test—a subjective analysis to determine whether substantial similarity exists between the two programs' expression.[74] The trier of fact evaluates the overall "look and feel" of the two programs,[75] subjectively comparing the programs rather than relying on the external criteria that marks the extrinsic test.[76] If the trier of fact finds similarities, the trier again analytically dissects the program into its elements to determine whether similarities result from unprotectable expression.[77]

C. Altai's *Three-Part "Abstraction-Filtration-Comparison" Test*

Rejecting *Whelan's* conceptual framework and discarding elements of the bifurcated test, the Second Circuit, in *Computer Associates International, Inc. v. Altai, Inc.*,[78] proposed a three-step abstraction-filtration-comparison procedure to determine substantial similarity between the nonliteral elements of two computer programs.[79] The *Altai* test breaks down the allegedly infringing program's structure into abstract levels.[80] The court does this by retracing the programmer's steps, starting with the program's most detailed level, usually

71. Sid & Marty Krofft Television Prod., Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977) (citing Arnstein v. Porter, 154 F.2d 464 (2d Cir. 1946) (original bifurcation of the substantial similarity test), *cert. denied,* 330 U.S. 851 (1947), and applying a two-part analysis). *See* Shaw v. Lindheim, 919 F.2d 1353, 1356–57 (9th Cir. 1990) (modifying the two-part analysis); Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475–77 (9th Cir.) (using the modified two-part analysis), *cert. denied,* 113 S. Ct. 198 (1992).

72. *Brown Bag,* 960 F.2d at 1475.

73. *See id.* at 1475–77.

74. *Id.* at 1475; *see also* Data East U.S.A., Inc. v. Epyx, Inc., 862 F.2d 204, 208 (9th Cir. 1988); *Krofft,* 562 F.2d at 1164–65.

75. *See Brown Bag,* 960 F.2d at 1476.

76. *Krofft,* 562 F.2d at 1164–65; *see also* Shaw v. Lindheim, 919 F.2d 1353, 1356 (9th Cir. 1990).

77. The court sifts out program elements where the idea and expression merge, where elements are *scenes a faire,* and where the expression is in the public domain. *See, e.g., Brown Bag,* 960 F.2d at 1473; *Data East,* 862 F.2d at 208; *see supra* notes 40–42 and accompanying text.

78. 982 F.2d 693 (2d Cir. 1992).

79. *See id.* at 706. This procedure incorporates a successive filtration methodology taken from scholarly commentary. *See* 3 NIMMER & NIMMER, *supra* note 43, § 13.03[F]; *see also* David Nimmer et al., *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases,* 20 ARIZ. ST. L.J. 625 (1988).

80. *See Altai,* 982 F.2d at 706–07.

consisting of the program's literal computer code.[81] Becoming increasingly more generalized, the court defines successive abstract levels of the program's structure, each level representing an unprotectable function that describes the structure within that abstract level.[82] The process ends when the court reaches the programmer's last step in creation, at which point the court is left with the program's most general abstraction level—the program's idea or purpose.[83] Throughout this "abstraction" process, the court filters out uncopyrightable elements of the computer program[84] by way of successive filtration.[85] After this filtration, the court compares the program's remaining copyrightable expression to that found in the allegedly infringing program to determine whether similarity exists, and if so, whether the similarity is significant to the owner's overall program.[86] In *Altai,* the court defined the copyrighted program's abstract levels as its object code, source code, parameter lists, macros, services required, and general outline.[87] The court subsequently filtered out the program's list of services and general outlines as being dictated by external factors[88] and found no substantial similarity between the competing programs' object code, source code, parameter lists, and macros.[89]

III. CRITIQUE OF EXISTING SUBSTANTIAL SIMILARITY TESTS FOR COMPUTER PROGRAMS

Existing substantial similarity tests fail to properly protect the nonliteral aspects of computer programs. Courts applying these substantial similarity tests fail to consider computer programs as a whole, and when comparing separate program elements do not use adequate dissection and filtration techniques. Because of this, courts have improperly defined the scope of copyright protection and incorrectly distinguished protectable expression. As a result, courts overlook programmers' original expression and erode copyright protection for their programs.

81. *Id.*

82. *Id.*

83. *See id.*

84. *Id.* at 707–10 (elements dictated by efficiency, elements dictated by external factors, and elements taken from the public domain); *see also supra* notes 40–42 and accompanying text.

85. *Altai,* 982 F.2d at 707; *see also* 3 NIMMER & NIMMER, *supra* note 43, § 13.03[F][5].

86. *Altai,* 982 F.2d at 710–11.

87. *Id.* at 714.

88. *Id.* at 715.

89. *Id.* at 714–15.

A. Whelan's *One-Step Integrated Test Is Too Broad and Fails to Consider the Program as a Whole*

Whelan's substantial similarity test[90] fails to properly protect nonliteral elements because it is too broad and because it does not consider the program as a whole. It is overbroad because it affords protection to unprotectable program elements. The court incorrectly assumes that only one idea underlies any computer program, and therefore isolates the program's main purpose and affords all other aspects of the program copyright protection. A program, however, is made up of smaller modules and subprograms, each of which may have its own idea and be unprotectable.[91] By erroneously defining computer structure, *Whelan* improperly extends copyright protection to unprotectable program elements. This results in a substantial similarity test that is too broad[92]—one that protects ideas as well as expression—and violates the Copyright Act's specific limitations.[93]

Additionally, the *Whelan* test fails to properly consider the program as a whole. The *Whelan* test extends protection to the separate, protectable nonliteral program elements.[94] By failing to view the program as a whole and to protect the program based on the coordination and arrangement of both protectable and unprotectable program elements, the *Whelan* test disregards the most creative aspects of computer programs.[95]

The deficiencies of the *Whelan* test's methodology become apparent when a court applies the test in the hypothetical KidPrint infringement suit. The court would first define KidPrint's purpose as the creation of an educational word processing program for children designed to teach the alphabet while expanding the child's vocabulary.[96] The court would then conclude that all program components not necessary for this purpose are protected.[97] Thus, an unprotectable idea such as

90. *See supra* notes 62–67 and accompanying text.

91. *See supra* notes 4–23 and accompanying text.

92. *See Altai,* 982 F.2d at 705–06 (criticizing *Whelan's* general formulation as being "descriptively inadequate" and as demonstrating "a flawed understanding of a computer program's method of operation"); 3 NIMMER & NIMMER, *supra* note 43, § 13.03[F], at 13-78.33, ("The crucial flaw in [*Whelan's*] reasoning is that it assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression."). Due to the heavy criticism generated by *Whelan's* program-idea concept, *Whelan's* correct emphasis on protecting nonliteral program structure has been largely overlooked.

93. *See supra* note 28 and accompanying text.

94. *See, e.g., supra* text accompanying notes 69–70.

95. *See supra* note 11 and accompanying text.

96. *See supra* note 65 and accompanying text.

97. *See supra* notes 65–66 and accompanying text.

KidPrint's "spell-checking" element would be deemed protectable program expression because it would not be found necessary to KidPrint's purpose. This would result in an expansive and unwarranted increase in the scope of copyright protection for KidPrint's separate components.[98] A court would find similarity between KidPrint's and KiddieType's data display modules based on subprogram and file structure interrelationship, but by failing to consider the unique selection, coordination, and arrangement of KidPrint's modules, subprograms, and file structures within the program as a whole, the court would deny copyright protection to KidPrint's overall program structure.[99]

B. The Ninth Circuit's Bifurcated Test Fails to Properly Examine Component Structure and to Consider the Program as a Whole

The Ninth Circuit's two-part substantial similarity test[100] fails to properly examine component structure and to consider computer programs as a whole. This test does not adequately protect nonliteral program aspects because the trier of fact must find similarity in the extrinsic stage before proceeding to determine whether substantial similarity of the "look and feel" of the program exists in the intrinsic stage.[101] In the extrinsic stage, the trier breaks down the two competing programs into elements for examination.[102] If the trier does not find similarity between the two programs' discrete elements, the trier never examines the relevant aspects of the copyrighted program's overall structure and component interrelationship.[103] The Ninth Circuit's two-part test therefore fails to consider the coordination and arrangement of the program as a whole and to recognize that copyright may exist based on the program's structure, apart from the protection given to its discrete program elements.[104]

98. KidPrint's hypothetical components include its four main modules: data entry, data display, spell-check, and print; the three primary subprograms making up KidPrint's data display module: animal picture display, animal sound production, character output to screen; and the file structures used in the storage and retrieval of each animal picture and sound used in the data display module.

99. *See, e.g., supra* text accompanying notes 68–70.

100. *See supra* notes 71–77 and accompanying text.

101. *See supra* notes 74–75 and accompanying text.

102. *See supra* note 73 and accompanying text.

103. *See supra* note 74 and accompanying text.

104. The petitioner in *Brown Bag* argued this on appeal, but the Ninth Circuit dismissed the contention due to evidence deficiency. Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1476 (9th Cir.), *cert. denied,* 113 S. Ct. 198 (1992).

Applying the Ninth Circuit's bifurcated substantial similarity test in the hypothetical KidPrint infringement suit, a court would not find substantial similarity between the KidPrint and KiddieType programs. The court would first analytically dissect the KidPrint and KiddieType programs into their respective components.[105] Objectively comparing individual components of the copyrighted program with those found in the allegedly infringing work,[106] the court would find no similarity of expression because each of KidPrint's discrete components, compared separately to those of KiddieType's, are dissimilar. Because no similarity exists in the extrinsic stage of the test, the court would end its analysis.[107] Had the court proceeded to the intrinsic stage, it would have found substantial similarity between the two programs based on the selection, coordination, and arrangement of KidPrint's program components and on the interrelationship between the subprograms and file structures within KidPrint's and KiddieType's data display modules. The test fails to properly consider the program as a whole and the structure within each component, and therefore fails to protect the original aspects of Compuhypo's KidPrint program.

C. Altai's *Three-Part Test Uses Inappropriate Analytical Techniques and Fails to Consider the Program as a Whole*

Altai's three-part abstraction-filtration-comparison test[108] uses inappropriate analytical techniques and fails to consider computer programs as a whole, thereby failing to protect nonliteral program aspects. To determine the protectable parts of the program, the *Altai* test breaks down the program into "abstract levels"[109] and filters out unprotectable program elements.[110] This dissection or abstraction process mischaracterizes computer components as abstract levels, however, causing an analysis of the wrong elements. The abstraction process also lacks definite limits on the extent of component abstraction. The filtration process fails because it prematurely sifts out unprotectable expression, neglecting potential similarity based on component structure.

105. *See supra* note 98.

106. *See supra* notes 72–73 and accompanying text.

107. *See supra* note 74 and accompanying text.

108. *See supra* notes 78–86 and accompanying text.

109. *See supra* notes 80–83 and accompanying text.

110. *See supra* notes 84–85 and accompanying text.

1. Altai's *Abstractions Process Is Ill-suited for Analytically Dissecting Computer Programs*

The abstraction process relies on a dissection method developed for traditional literary works.[111] This method of analysis is wholly inadequate when applied to computer programs because the individual elements of traditional literary works and computer programs are different in both their definition and their interrelationship.[112] Although the literal elements of traditional literary works and computer programs are similar, both composed of fixed written words and symbols, the nonliteral elements of each are not analogous. Traditional literary works such as novels and plays have readily definable nonliteral elements such as themes, characters, plots, and sequences of events.[113] A computer program's nonliteral elements are less defined and include a wide range of program elements, from screen outputs to parameter lists.[114] Because the nonliteral elements of each are so different, an abstraction process designed for traditional literary works inadequately protects computer programs.

Additionally, courts applying the abstraction method to computer programs mischaracterize computer elements as "abstract levels."[115] When breaking down the program into abstract levels, the *Altai* test separates the program into its object code, source code, parameter lists, macros, services required, and general outline.[116] As abstract levels representing separate functions or ideas, these elements are uncopyrightable.[117] Some of these "abstract levels," however, represent elements within a computer program rather than abstract levels analogous to those found within traditional literary works, and may be copyrightable as original expression. For example, under the *Altai*

111. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706–07 (2d Cir. 1992). *Altai's* abstraction analysis is based on the Second Circuit's enunciation of Judge Learned Hand's abstractions test, originally created for and applied to literary works such as novels and plays. *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930), *cert. denied,* 282 U.S. 902 (1931).

112. *See* LEAFFER, *supra* note 43, § 9.5[E] (suggesting that applying copyright law to computer programs as literary works is problematic); *see also supra* note 54 and accompanying text.

113. Stewart v. Abend, 495 U.S. 207, 238 (1990) (recognizing that copyrightable nonliteral elements of a book include its unique setting, characters, plot, and sequence of events); Shaw v. Lindheim, 919 F.2d 1353, 1356–57 (9th Cir. 1990) (suggesting that traditional literary works can be broken down into standard and generally accepted nonliteral elements such as plot, themes, mood, setting, pace, characters, and sequence of events); *Nichols,* 45 F.2d at 121 (recognizing that protectable nonliteral elements of a play include characters, sequence of incidents, and plot).

114. *See supra* notes 17–22 and accompanying text.

115. *See supra* notes 80–83 and accompanying text.

116. *See supra* notes 80–89 and accompanying text.

117. *See supra* note 82 and accompanying text.

test, a court may define a parameter list[118] as an abstract layer that is an unprotectable idea. A parameter list, however, may be a protectable expression of an idea if analyzed as a program element. Thus, by mischaracterizing computer elements as abstract levels, the court denies protection to a programmer's original nonliteral program elements.

Furthermore, in applying the abstractions test, the *Altai* court fails to limit the extent of element abstraction. The *Altai* court relies on the abstraction process to break down the computer program into separate elements.[119] By not providing a limit to the program abstraction, however, a court may eventually abstract the program into pieces so elemental that originality no longer exists. At that point, a program's expression would merge with the idea it represents and a court would eliminate all copyrightability.[120]

2. Altai's *Filtering Procedure Prematurely Diminishes Legitimate Copyright Protection*

After abstracting the computer program into elemental parts, the *Altai* test separates the unprotectable elements from the protectable elements to allow the trier of fact to determine substantial similarity based solely on similarity between protectable expression.[121] This filtration process sifts out elements of computer programs to which the merger,[122] *scenes a faire,*[123] and public domain[124] copyright doctrines deny protection.[125] This filtration procedure, however, prematurely separates unprotectable program elements from their related and potentially protectable components.[126] A program's creativity results

118. *See supra* note 20 and accompanying text.

119. *See supra* notes 80–83 and accompanying text.

120. This process resembles abstracting the text of a book into its conceptual elements. While at the level of a complete book, chapter, paragraph, or even sentence, the material may be sufficiently original to retain its copyright protection. If abstraction of these elements continues, the court will abstract the original expression of words to an outline of the text, and then to the concept or idea that motivated the literal expression. At this abstract level the work's expression has merged with its idea and all copyright protection is lost. *Cf.* David Bender, Computer Associates v. Altai: *Rationality Prevails,* COMPUTER LAW., Aug. 1992, at 5–6 (suggesting that "[i]f one breaks [program] expression into small enough chunks, each chunk will be so elemental that it will effectively merge with the idea it represents").

121. *See supra* note 86 and accompanying text.

122. See *supra* note 40 for a description of the merger doctrine.

123. See *supra* note 41 for a description of the *scenes a faire* doctrine.

124. See *supra* note 42 for a description of the public domain doctrine.

125. *See supra* note 84 and accompanying text.

126. The Ninth Circuit finds this filtration procedure premature, and instead filters out unprotectable elements after determining similarity. *See* Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1476 (9th Cir.) ("[W]here two works are found to be similar without

primarily from the coordination between and interrelationship among its components.[127] Unprotectable elements may interact with each other in a creative way, just as protectable and unprotectable elements may interact with each other in a creative way.[128] By filtering out unprotectable elements before comparison for similarity, the *Altai* test denies protection to the program as a whole based on the selection, coordination, and arrangement of its protectable and unprotectable components as well as on each component's separate structure. Thus, *Altai's* test prematurely filters out program elements and inadequately protects a programmer's originality.[129]

The *Altai* three-part substantial similarity test would deny protection to Compuhypo's KidPrint program. Applying this test, a court would first dissect KidPrint into abstract levels by retracing Compuhypo's development steps in reverse order.[130] KidPrint would be broken down into its four main modules,[131] and its data display module would be abstracted into its animal picture display, animal sound production, and character output to screen subprograms along with its corresponding file structures. A court, examining KidPrint's discrete program components, would filter each of them out as unprotectable expression.[132] KidPrint's copyrightable expression, however, is based on the original selection, coordination, arrangement and interrelationship of its unprotectable program components. Thus, filtering out KidPrint's program components before comparison would leave the court with no expression to compare to KiddieType. Under this test, the court would not find copyright infringement, and would effectively deny copyright to KidPrint's original nonliteral program aspects.

---

regard to the scope of the copyright in the plaintiff's work . . . the source of the similarity must be identified and a determination made as to whether this source is covered by plaintiff's copyright."), *cert. denied,* 113 S. Ct. 198 (1992).

127. *See supra* note 11 and accompanying text.

128. *See supra* notes 5–11 and accompanying text.

129. *See* Gates Rubber Co. v. Bando Am., Inc., 798 F. Supp. 1499, 1516 (D. Colo. 1992) (rejecting the use of the filtration test prior to comparison because such a procedure "has the real potential to eviscerate the application of [the substantial similarity test]"); Anthony L. Clapes & Jennifer M. Daniels, *Revenge of the Luddites: A Closer Look at* Computer Associates v. Altai, COMPUTER LAW., Nov. 1992, at 16 ("Filtering ideas out of each level of abstraction in a program . . . deprives the work of its essence.").

130. *See supra* notes 80–83 and accompanying text.

131. *See supra* note 98.

132. *See supra* note 84 and accompanying text.

IV. COURTS SHOULD ADOPT A TWO-TIERED SUBSTANTIAL SIMILARITY TEST

To rectify the problems with existing substantial similarity examinations, courts should adopt a two-tiered substantial similarity test that considers computer programs both as an integrated whole and as a set of separate components. This test focuses the court's attention on a computer program's nonliteral aspects—where the majority of a programmer's creative input exists.[133] Upon finding similarity at either tier, the court should determine the substantiality of that similarity and conclude its infringement analysis. This test would better determine, compare, and protect the original nonliteral aspects of computer programs.

A. Tier One: Program-as-a-Whole Analysis

In the first tier of analysis, courts should compare competing programs as complete, integrated works. At this stage, the programs would be viewed as comprised of both protectable and unprotectable elements. The court should consider the programs as a whole because doing so recognizes the compilation-type nature of computer programs and extends copyright protection to the creative aspects of computer programs.

The court should examine the programs as a whole to determine similarity in selection, coordination, and arrangement of the program components as courts do when analyzing compilations. For purposes of this part of the test, courts should analogize computer programs to traditional compilations such as legal casebooks to consider programs as a whole. A typical casebook is composed of both unprotectable elements (cases) and protectable elements (original commentary). An editor chooses which cases to include, what order to place them in, and how to most effectively arrange them. Due to the public nature of court decisions, copyright does not protect these cases individually. Copyright protection may exist, however, for the independent original commentary and for the work as a whole, so long as its selection, coordination, or arrangement is original.[134]

133. *See supra* note 11 and accompanying text.

134. *See, e.g.*, Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 556–57 (1985) (holding that President Ford could not prevent others from copying the unprotectable historical facts of his work but could prevent others from copying his original and "subjective" descriptions associated with these facts); *cf.* Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc., 111 S. Ct. 1282, 1289 (1991) (denying copyright protection for a phone book as a compilation work because the court found no original selection and arrangement).

Like compilations, computer programs are often an integration of both unprotectable and protectable elements.[135] Furthermore, as with compilations, most of the programmer's original expression exists in the selection, coordination, and arrangement of computer program components.[136] Because computer programs are similar to compilations[137] and because a compilation-type analysis would adequately protect the program's nonliteral structure, courts should view programs as a whole to determine substantial similarity, considering the selection, coordination, and arrangement of program components.[138] In viewing the program as a whole, courts would examine the program's dynamic behavior—the way the program responds to input from its human user and from other program components—and, if applicable, the program's audiovisual representations, using any similarities as evidence of copying of the program structure.[139] Using this approach, courts would properly incorporate an author's creativity

135. *See* DALE & LILLY, *supra* note 1, at 3 (describing how programmers often rely upon a "shared body of knowledge" that has been collected over time in program creation); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs,* 41 STAN. L. REV. 1045, 1057 (1989) (suggesting that programmers draw heavily upon an existing knowledge base to create their programs, often using elements of existing programs).

136. *See supra* note 11 and accompanying text.

137. The description of compilations in the 1976 original House report further supports the compilation analogy: "A 'compilation' results from a process of selecting, bringing together, organizing, and arranging previously existing material of all kinds, regardless of whether the individual items in the material have been or ever could have been subject to copyright." HOUSE REPORT, *supra* note 45, at 5670.

138. Clapes & Daniels, *supra* note 129, at 13 ("Even where . . . elements [of computer programs] are individually unprotected by copyright . . . , the selection, arrangement and/or organization of such unprotected elements may be sufficiently original to be protected by copyright."); *see also* Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 840–41 (Fed. Cir. 1992) (concluding that the unique selection and arrangement of computer program expression that generates a data stream is sufficiently creative to be protected under copyright); *cf.* Atari Games Corp. v. Oman, 979 F.2d 242 (D.C. Cir. 1992) (concluding that audiovisual works are analogous to compilations of fact, and that copyright protection is available to audiovisual works when viewed as a whole based on the author's selection and arrangement of individual elements).

139. The way that traditional compilations are organized and presented affects the way they are used. Students, for example, are guided in their study of the law by the order and manner in which their casebooks present cases and commentary, thereby making casebooks distinguishable based on their organization and presentation. In an analogous fashion, courts can determine similarity between computer programs based in part on how each program interacts with and is presented to its user—namely its dynamic behavior and audiovisual representations. *See* Gates Rubber Co. v. Bando Am., Inc., 798 F. Supp. 1499, 1518–19 (D. Colo. 1992) (holding that a program's behavior is significant in determining computer program copyright infringement); Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1244 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987) (concluding that audiovisual representations are inherently related to the underlying program that generates them and that courts should give probative value to the resulting similarity between audiovisual displays).

into the substantial similarity analysis when considering programs' nonliteral structure, thereby adequately protecting programmers' original expression.

B. Tier Two: Program-Component Analysis

Regardless of similarity between the programs viewed as a whole in the first tier, courts should proceed to tier two, the program-component analysis. At this stage, courts dissect the competing programs into their respective components, breaking them down according to their corresponding modules, subprograms, and file structures.[140] Courts should then examine the literal and nonliteral aspects or elements associated with each of these components,[141] comparing the elements of the copyrighted program's components to those of the allegedly infringing program's components. If the court finds similarity between competing program components, it should sift out the unprotectable elements of the copyrighted program's components,[142] leaving only the protectable "core" of similar expression. If the similarity found at either tier constitutes a significant portion of the copyrighted program—examining the quality as well as the quantity of the similarity—the court would find substantiality exists.[143]

Applying the proposed two-tiered test to the KidPrint program, a court would properly find similarity between KidPrint and KiddieType. In the program-as-a-whole tier, the court would recognize the originality involved in selecting, coordinating, and arranging both protectable and unprotectable components in KidPrint. Comparing KidPrint's and KiddieType's program structure, the court would examine the dynamic behavior of the two programs. The court would find similarity between how both KidPrint and KiddieType input data from the user, send data to and from each module, subprogram, and file structure, and ultimately display the data on the screen. Examining the resulting audiovisual representations as evidence of copying, the court would again find similarity based on how the data and related animal pictures and sounds appear to the user.

In the program-component tier, the court would find similarity based on the interrelationship between related subprograms and file structures. Initially, the court would dissect KidPrint and KiddieType into their respective components, namely their modules, sub-

140. *See supra* text accompanying notes 7–9.

141. *See supra* notes 16–22 and accompanying text.

142. *See supra* notes 40–42 and accompanying text.

143. *See, e.g., Whelan,* 797 F.2d at 1245; *see also* 3 NIMMER & NIMMER, *supra* note 43, § 13.03[A].

programs, and file structures.[144] Next, the court would compare the elements of each program's component structure, finding no similarity between the discrete program components. In comparing the component structure of KidPrint's data display module to KiddieType's, however, the court would find similarity based on the interrelationship among the modules' three subprograms (animal picture display, animal sound production, and character output to screen) and with the animal picture and sound file structures. The court would then sift out elements of KidPrint's program components that are unprotectable due to the merger, *scenes a faire,* and public domain doctrines,[145] leaving the interrelationship between subprograms and file structures within KidPrint's data display module as protectable expression. Ultimately, the court would conclude that the similarities between the programs as a whole and between the structure of the programs' data display modules—because it is essentially the heart of KidPrint's creative aspects—are sufficiently substantial to justify a finding of copyright infringement.

V. CONCLUSION

To protect programmers' rights in their original works, Congress chose not to create a separate right for computer programs, but to extend existing copyright principles to programs. This decision left courts forcing the proverbial square peg into a round hole.[146] By not considering programs as a whole and by using inappropriate analytical techniques to determine infringement, courts fail to recognize and properly protect programmers' originality in the nonliteral aspects of their computer programs.

Under existing substantial similarity tests, illicit copying of nonliteral aspects, as illustrated in the hypothetical infringement suit, will go unpunished. Courts should adopt a two-tiered test that considers computer programs both as an integrated whole and as separate components. Designed to recognize the original aspects of computer programs, this test would adequately protect programmers' rights in their programs.

144. *See supra* note 98. *See generally supra* notes 7–9.

145. *See generally supra* notes 40–42.

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