Georgia State University Law Review

Volume 14
Issue 2 *March 1998*

Article 3

3-1-1998

The Feasibility of the Abstraction-Filtration-Comparison Test for Computer Software Copyrightability (And Analysis of Bateman v. Mnemonics)

Lisa M. Gable

THE FEASIBILITY OF THE ABSTRACTION-FILTRATION-COMPARISON TEST FOR COMPUTER SOFTWARE COPYRIGHTABILITY (AND ANALYSIS OF *BATEMAN V. MNEMONICS*)

INTRODUCTION

Developing intellectual property protection for "rapidly changing, high-technology works is perhaps modern copyright law's most difficult task."[1] The computer software industry currently loses over one billion dollars annually due to unauthorized copying of software in the United States alone.[2]

The computer software industry deems copyrights the best way to protect intellectual property rights for computer programs.[3] Courts agree that the literal and nonliteral[4] elements of computer programs are protectable under copyright law.[5] However, courts are experiencing "a state of creative ferment concerning the methods by which nonliteral elements of computer programs may be identified and analyzed for copyrightability."[6]

Several circuits use the abstraction-filtration-comparison (AFC)[7] test or similar methods as the leading test for determining software copyrightability.[8] The AFC test ascertains whether "substantial similarity"[9] exists between plaintiff's and

---

1. Jon S. Wilkins, Note, *Protecting Computer Programs as Compilations Under* Computer Associates v. Altai, 104 YALE L.J. 435, 469 (1994).

2. *See* Himanshu S. Amin, *The Lack of Protection Afforded Software Under the Current Intellectual Property Laws*, 43 CLEV. ST. L. REV. 19, 20 (1995). International piracy costs $8 to $10 billion in lost revenues each year. *See id.*

3. *See id.* at 30.

4. *See infra* Part I.C. (discussing literal and nonliteral elements of computer programs).

5. *See* Michael MacAdam Barry, Note, *Software Copyright Upgrade*—Engineering Dynamics v. Structural Software *Extends Abstraction-Filtration-Comparison to Software Input Data Formats*, 39 ST. LOUIS U. L.J. 1309, 1320-21 (1995).

6. Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1341 (5th Cir. 1994).

7. See *infra* Part II for a thorough discussion of the history and current status of the AFC test.

8. See *infra* Part II.A. for a thorough discussion of decisions within the circuits, either adopting or rejecting the AFC test, or using similar traditional copyright doctrines to determine software copyrightability.

9. See *infra* Part I.D. for a discussion of substantial similarity in proving copyright infringement.

defendant's computer program structures in three distinct steps—abstraction, filtration, and comparison.[10] In *Bateman v. Mnemonics, Inc.*,[11] the Eleventh Circuit adopted AFC testing for determining software copyrightability.[12]

Part I of this Comment discusses the basics of copyright law as applied to computer programs. Part II examines the history of the AFC test and the disagreement regarding the test among the circuits. Part III looks at AFC testing as applied by the court in *Bateman*. Part IV analyzes the court's holding in *Bateman*, and Part V considers whether AFC testing is viable for protecting computer software.

I. THE BASICS OF COPYRIGHT LAW AS APPLIED TO COMPUTER SOFTWARE

A. *The Constitution and the Copyright Act*

Copyrights derive from the United States Constitution: "The Congress shall have Power . . . [t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries[.]"[13] Section 102 of the Copyright Act of 1976 (Copyright Act) defines the subject matter of copyright: "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."[14]

To be protected by copyright, a work of authorship must be original, meaning that the author must engage in an "intellectual endeavor"[15] and express a "minimal amount of creativity."[16] To protect the goal of copyright law, which seeks to allow the public to benefit from an author's work, a court must not be more

10. *See* Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992).
11. 79 F.3d 1532 (11th Cir. 1996).
12. *See generally id.* at 1536, 1545; Bruce G. Joseph & David A. Vogel, *Copyright Protection of Software and Compilations: A Review of Critical Developments 1991-1996*, 441 PRACTISING L. INST./PATENTS, COPYRIGHTS 369, 412 (1996).
13. U.S. CONST. art. I, § 8, cl. 8; Barry, *supra* note 5, at 1315.
14. 17 U.S.C. § 102(a) (1994).
15. Amin, *supra* note 2, at 31.
16. *Id.; see also* Adam E. McKinney, Comment, *Copyright Protection for Functional Works: Where Does the Fifth Circuit Draw the Line Between Idea and Expression?*, 47 BAYLOR L. REV. 249, 252 (1995).

concerned with the author's monetary compensation or just reward than the public benefit.[17]

B. The Copyright Act as Applied to Computer Software

In 1980, Congress amended the Copyright Act to extend copyright protection to computer software.[18] 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."[19] Courts now must apply the definition of literary works of authorship to software.[20] " '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, phonorecords, film, tapes, disks, or cards, in which they are embodied."[21]

C. Elements of Computer Programs

To further understand why courts and Congress classify computer programs as literary works, one must understand the composition of software, which courts divide into literal and nonliteral elements. Programmers write software in a language known as "source code."[22] Source code must be translated into object code, a language the computer understands, which consists of binary ones and zeros.[23] The source and object codes compose the literal elements of a computer program.[24]

Courts label the nonliteral elements of a program in various ways, including "structure, sequence and organization" or "look and feel."[25] The nonliteral elements of a program can be

---

17. *See* William F. Porter, *Breaking the Silence of a Divided Court: An Analysis of the First Circuit's Decision in* Lotus v. Borland, 36 IDEA 273, 277 (1996).

18. *See* Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3028; Barry, *supra* note 5, at 1315.

19. 17 U.S.C. § 101 (1994).

20. *See* Barry, *supra* note 5, at 1316.

21. 17 U.S.C. § 101 (1994).

22. Amin, *supra* note 2, at 21.

23. *See id.*

24. *See generally* Bateman v. Mnemonics, 79 F.3d 1532, 1543 n.25 (11th Cir. 1996); Barry, *supra* note 5, at 1318-19.

25. Barry, *supra* note 5, at 1319 (quoting Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527, 536 (5th Cir. 1994); Apple Computer, Inc. v. Microsoft Corp, 799 F. Supp. 1006, 1016 (N.D. Cal. 1992)).

analogized to the characters and plot in a novel or the structure and organization of a textbook.[26] "[L]imiting copyright protection to the literal series of words in a story would allow appropriation of the expression of the story by merely rearranging and substituting words without changing the story's meaning."[27]

The Copyright Act does not mention nonliteral elements of computer programs or user interfaces in its language.[28] A computer programmer can rewrite code in one program so that it is literally dissimilar to another program, yet the results of both programs' instructions to the computer will be indistinguishable.[29] Therefore, "[l]imiting copyright protection to the literal software belies the true nature of software because nonliteral aspects represent original and creative expression of the author."[30]

D. Proving Copyright Infringement

To prevail in a copyright infringement action, a plaintiff must show ownership of a valid copyright and that the defendant copied the plaintiff's work.[31] A plaintiff may show the defendant copied by proving that the defendant had access to plaintiff's work and that the defendant's work is substantially similar to plaintiff's work.[32]

The court must use traditional copyright doctrines[33] to determine what elements of a plaintiff's work are protectable.[34] Elements of a work, such as a play, include selection and arrangement, plot sequence, character development, and

---

26. *See* Linda Skon, Comment, *Copyright Protection of Computer User Interfaces: "Creative Ferment" in the Courts*, 27 ARIZ. ST. L.J. 1063, 1065 (1995).

27. Barry, *supra* note 5, at 1319.

28. *See* 17 U.S.C. § 101 (1994); Skon, *supra* note 26, at 1065. "A 'computer user interface' is the way the user communicates with a computer program. User interfaces may include elements of a program's visual display, the program's commands that allow users to control a computer's operations, and the relationship between commands and display elements." Skon, *supra* note 26, at 1063.

29. *See* Barry, *supra* note 5, at 1320.

30. *Id.* at 1319.

31. *See id.* at 1317; *see also* Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340 (1991).

32. *See* Barry, *supra* note 5, at 1317 (quoting *Altai*, 982 F.2d at 701).

33. See *infra* Part I.E. for a discussion of traditional copyright doctrines that limit copyright protection, including idea/expression merger, the useful article doctrine, *scenes a faire*, public domain, and fair use.

34. *See* Wilkins, *supra* note 1, at 439.

particular dialogue.[35] Once the court ascertains the protectable expression of a plaintiff's work, the court must compare that expression to the defendant's work to determine if a substantial similarity exists between the two works.[36]

Exactly how much of a work must be copied for substantial similarity to exist depends upon the type of work examined.[37] "For example, the content of a factual compilation such as a phone directory must be virtually identical to that of the copyrighted work in order to constitute infringement. In contrast . . . copying even a single line of a poem may constitute infringement."[38] The phone book example is a quantitative assessment of substantial similarity, whereas the poem example is a qualitative assessment.[39]

E. Limiting Copyright Doctrines

Several traditional copyright doctrines limit or even deny protection to works of authorship.[40] These doctrines include the idea/expression merger, the useful article doctrine, *scenes a faire*, public domain, and fair use.[41]

1. Idea/Expression Merger

In looking at idea/expression merger, it is helpful to remember that the purpose of copyright law is to protect idea expression, not the idea itself.[42] The scope of copyright does not provide protection for "any idea, procedure, process, system, method of operation, concept, principle, or discovery[.]"[43]

The concept of idea/expression merger derived from the very early case of *Baker v. Selden*,[44] in which the Supreme Court held that the elements of a work that were necessary to use the work's underlying idea were not protectable under copyright

35. *See id.* at 436, 438.
36. *See id.* at 439.
37. *See id.*
38. *Id.*
39. *See id.* An "ad hoc value judgment of whether the copying is *qualitatively* significant to the copyrighted work" offers only general guidance in determining substantial similarity. *Id.*
40. *See* Barry, *supra* note 5, at 1323.
41. *See generally id.*
42. *See* Amin, *supra* note 2, at 31.
43. 17 U.S.C. § 102(b) (1994).
44. 101 U.S. 99 (1879).

law.[45] In *Baker*, the defendant used an accounting system, as explained and illustrated in the plaintiff's book on accounting, to make improvements and additions to his own accounting system.[46] The Court held that the plaintiff could not secure a copyright for the "use of the system or method of book-keeping [sic]" described in the plaintiff's book.[47] The Court explained that "[t]he very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book."[48] The Court also wrote that "explanation" is the territory of copyright, and "use" is the territory of patent.[49]

"[C]opyright cannot directly or indirectly prevent the use of unprotected ideas of utilitarian features of functional works."[50] The determination of whether idea and expression are merged requires a decision whether the idea can be expressed in various ways.[51] This determination is quite difficult and even more so when a functional work is being evaluated for protection.[52]

2. *The Useful Article Doctrine/Functional Instruments*

Computers represent a "twist in the body of copyright law"[53] because they may be defined both as written expression and functional instruments.[54] A tension exists between patent law and copyright law in evaluating computer software.[55] A patent protects the "useful arts" by giving a monopoly over a "process, machine, manufacture, or composition of matter[,]"[56] and it stringently requires originality.[57] On the other hand, copyright extends protections to software in the same manner as that extended to "books, plays, and musical recordings[,]"[58] protecting

45. *See id.* at 104; Skon, *supra* note 26, at 1066.
46. *See Baker*, 101 U.S. at 100, 101.
47. *Id.* at 101.
48. *Id.* at 103.
49. *Id.* at 105.
50. Skon, *supra* note 26, at 1067.
51. *See* Porter, *supra* note 17, at 279.
52. *See* McKinney, *supra* note 16, at 251.
53. Porter, *supra* note 17, at 279.
54. *Id.*
55. *See* McKinney, *supra* note 16, at 251.
56. *Id.* (quoting 35 U.S.C. § 101 (1988)).
57. *Id.* at 251, 252.
58. Amin, *supra* note 2, at 33.

the form of the expression, but not the underlying idea or process expressed. Patent theory proponents argue that a work's "[f]unctionality and ease of use are far more important to a purchaser of software than its aesthetic appeal or originality."[59] Copyright for useful articles provides protection only for designs existing separately from the utilitarian aspects of the work.[60] A functional work's purpose is solely utilitarian, whereas a nonfunctional work's purpose is purely aesthetic.[61]

Congress defined a "useful article" as "an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information."[62] If a work meets the definition of a useful article, a copyright may only be had for "designs that can exist separately from the utilitarian aspects of the work."[63] Applying the useful article doctrine to any work of authorship under the Copyright Act will exclude the functional aspects of that work; only the implementations of functions would be protected.[64] Further, the useful article doctrine does not apply to literary works; a strict reading of the statute applies the doctrine only to pictorial, graphic, or sculptural works.[65]

If courts grant too much protection to computer programs as utilitarian articles, public access to new software innovations will be limited because computer companies will need more time, money, and effort to create a different expression for each new product.[66] Courts must scrutinize functional works closely so that companies do not gain monopolies over ideas without meeting the requirements of patent law.[67] One commentator has stated that the AFC test, which takes into account utilitarian aspects of computer programs, was developed to help decipher

---

59. *Id.*
60. *See* Skon, *supra* note 26, at 1067.
61. *See* McKinney, *supra* note 16, at 251.
62. 17 U.S.C. § 101 (1994).
63. Skon, *supra* note 26, at 1067.
64. *See* Dennis M. Carleton, Comment, *A Behavior-Based Model for Determining Software Copyright Infringement*, 10 HIGH TECH. L.J. 405, 419 (1995).
65. *See id.* (citing Jack E. Brown, *"Analytical Dissection" of Computer Software—Complicating the Simple and Confounding the Complex*, 25 ARIZ. ST. L.J. 801, 833 (1993) ("[D]iscussing how the term 'useful article' only appears in §§ 101 and 113 of the Copyright Act, which pertain to the scope of protection for pictorial, graphic, or sculptural works.")).
66. *See* Skon, *supra* note 26, at 1082.
67. *See* McKinney, *supra* note 16, at 251.

the copyrightability of works that are not purely utilitarian or aesthetic in nature.[68]

3. Scenes a faire

Courts use *scenes a faire* as a protection limiting doctrine.[69] Under *scenes a faire*, verbatim copying does not infringe if the copied expression consists of "stock scenes or scenes that flow necessarily from common unprotectable ideas."[70] For example, "it is virtually impossible to write about a particular historical era or fictional theme without employing certain 'stock' or standard literary devices."[71]

4. *Public Domain*

Copyright does not protect expression if that expression is in the public domain.[72] Anything in the public domain, such as a work whose copyright has expired, is "free for the taking and cannot be appropriated by a single author even though it is included in a copyrighted work."[73]

5. *Fair Use*

Defendants use the fair use doctrine to affirmatively excuse copyright infringement.[74] The fair use of a copyrighted work "for purposes such as criticism, comment, news reporting, teaching . . ., scholarship, or research, is not an infringement of copyright."[75] The fair use limitation "requires courts to make subjective assessments of the purpose of the use, the nature of the copyrighted work, the relative amount of copying and the effect of the use on the market for or value of the copyrighted work."[76]

The doctrine of fair use comes into play when considering compatibility between computer programs. In order for a

---

68. *See id.*
69. *See* Barry, *supra* note 5, at 1324.
70. *Id.* (quoting See v. Durang, 711 F.2d 141, 143 (9th Cir. 1983)).
71. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 709 (2d Cir. 1992) (quoting Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980)).
72. *See Altai*, 982 F.2d at 710.
73. *Id.*
74. *See* Barry, *supra* note 5, at 1325.
75. 17 U.S.C. § 107 (1994).
76. Barry, *supra* note 5, at 1325; *see also* 17 U.S.C. § 107 (1994).

computer program to be compatible with an entire computer system, the "communication must be unimpeded throughout the system."[77] The software marketplace requires compatibility because it increases the number of users and available applications.[78] The marketplace isolates software if the software lacks compatibility with other programs.[79] Software companies spend much time, research, and development in making programs compatible.[80]

Companies create compatibility using reverse engineering.[81] "Through this process [of reverse engineering], computer programmers 'disassemble' the program to determine how it functions. The programmers then use the information they discover during that process to develop new programs which render their products compatible."[82] Unprotected portions of computer code, which are important for other companies to use and build upon, may be surrounded by copyrighted code, attainable only by reverse engineering.[83]

Two of the main cases in the area of fair use and reverse engineering are *Sega Enterprises Ltd. v. Accolade, Inc.*[84] and *Atari Games Corp. v. Nintendo of America, Inc.*[85] In *Atari*, the Federal Circuit concluded that using reverse engineering to discern the unprotectable ideas in the object code of a computer program is a fair use for compatibility reasons.[86] In *Sega*, the Ninth Circuit held that disassembly constitutes a fair use when it is the only way to access the ideas and functional elements in a copyrighted computer program to make other programs compatible or to create compatible software.[87]

---

77. Timothy S. Teter, Note, *Merger and the Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases*, 45 STAN. L. REV. 1061, 1063 (1993).

78. *See* Karen E. Georgenson, Comment, *Reverse Engineering of Copyrighted Software: Fair Use or Misuse?*, 5 ALB. L.J. SCI. & TECH. 291, 291 (1996).

79. *See id.* at 292.

80. *See id.*

81. *See id.*

82. *Id.*

83. *See id.* at 293. See also *id.* at 294-97, for a brief but concise discussion of the reverse engineering process.

84. 977 F.2d 1510 (9th Cir. 1992).

85. 975 F.2d 832 (Fed. Cir. 1992).

86. *See* Robert G. Gurrola, *Software Infringement Analysis Must Include Literal Similarities, Compatibility Defense, 11th Circuit Rules*, 1996 WLN 259531, Apr. 4, 1996 (available in West's Legal News 2831, Intellectual Property: Copyright Library File).

87. *See id.*; Georgenson, *supra* note 78, at 308 (arguing that consistent application

The holdings in *Sega* and *Atari* indicate that due to monopoly and innovation problems, "not protecting elements necessary for compatibility, while imperfect, is preferable to overbroad, long-lived protection."[88] Network externalities[89] may give a monopoly when compatibility is not permitted as a fair use.[90] Further, lack of compatibility affects innovation because people will not buy new software that is incompatible, and incentives to create new programs would cease to exist.[91] One commentator has urged that elements dictated by efficiency, internal interface elements required to achieve compatibility, and elements of user interfaces that are de facto interface standards should be unprotected by copyright.[92]

II. HISTORY OF THE AFC TEST AND THE SPLIT AMONG THE CIRCUIT COURTS

The circuit courts are "far from unanimous on how to protect software,"[93] but the *Computer Associates International v. Altai, Inc.* AFC test is the leading method.[94] Most circuits have adopted the AFC testing method but have reached different results with regard to protection for computer software.[95] The split result among the circuits indicates that courts continue to

---

of fair use in reverse engineering cases has not happened since *Atari* and *Sega* were decided because courts rely on public policy considerations rather than relying on the statutory factors). "[T]he courts seem to go through the motions of applying the statutory factors and then disregard the results when they are contrary to a finding of fair use." Georgenson, *supra* note 78, at 308; *see also id.* at 312-20 (urging use of copyright misuse defense instead of fair use defense because copyright misuse permits courts to address public policy, which is true basis of courts' conclusions, rather than inconsistently analyzing statutory fair use factors).

88. Teter, *supra* note 77, at 1070.

89. Network externalities in the context of compatibility of computer software may best be described by an example. "For example, compatibility encourages the formation of networks, through which users can exchange files. Standardization . . . prevents user 'lock-in' because users do not have to learn a new user interface in order to switch application programs." Teter, *supra* note 77, at 1066-67; *see also id.* at 1066-70 (discussing how network externalities are affected by compatibility (or lack thereof)).

90. *See id.* at 1067.

91. *See id.*

92. *See id.* at 1070 (discussing these elements further).

93. Wilkins, *supra* note 1, at 441.

94. *See id.*

95. *See* Lisa T. Oratz, *User Interfaces: Copyright vs. Trade Dress Protection*, 13 COMPUTER LAW. 1, 3 (1996).

grapple with exactly how far protection for nonliteral elements of software should extend.[96]

A. *Case History*

1. *A Very Early Beginning:* Nichols v. Universal Pictures Corp.

The AFC test originated in *Nichols v. Universal Pictures Corp.*[97] There, Judge Learned Hand decided that a play had not infringed the copyright of an earlier play.[98] Judge Hand stated that "[i]t is of course essential to any protection of literary property, whether at common-law or under the statute, that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations."[99] Judge Hand further stated that a determination of substantial similarity is more difficult when a plagiarist takes "an abstract of the whole."[100]

> Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended.[101]

The court held that the plot of the second story was too different to infringe and that much of the content of the play was in the public domain anyway and thus unprotectable.[102] Several courts have cited *Nichols* in adopting or discussing the AFC test because of Judge Hand's discussion of using "series of abstractions" to discern copyrightability.[103]

---

96. *See* Porter, *supra* note 17, at 291.

97. 45 F.2d 119 (2d Cir. 1930); *see* Dennis M. Carleton, Note, Lotus Development v. Borland International*: Determining Software Copyright Infringement is not as Easy as 1-2-3*, 56 U. PITT. L. REV. 919, 923 (1995).

98. *Nichols*, 45 F.2d at 121.

99. *Id.*

100. *Id.*

101. *Id.*

102. *See id.*

103. *See, e.g.*, Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1543 (11th Cir. 1996); Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1343 (5th Cir.

2. *AFC Test Development Continues:* Apple Computer, Inc. v. Franklin Computer Corp.

For many years after *Nichols*, AFC testing continued to develop through traditional copyright doctrines. Although the Third Circuit in *Apple Computer, Inc. v. Franklin Computer Corp.*[104] did not explicitly use an AFC test analysis to determine copyrightability, the court did use traditional copyright doctrines to determine copyrightability of the program involved in that case, including idea/expression merger and the useful article doctrine.[105]

Apple sued Franklin after Franklin copied Apple's operating system programs to achieve compatibility with Franklin's personal computer.[106] Franklin did not dispute that it copied Apple's programs, but defended that it wanted to achieve compatibility, and argued that the Apple programs contained no copyrightable subject matter.[107] The court, in this threshold, landmark decision, held that a "computer program, whether in object code or source code, is a 'literary work' and is protected from unauthorized copying, whether from its object or source code version."[108] The court also found that a program embedded in a ROM chip[109] and computer operating system programs are copyrightable.[110] The Third Circuit held that computer operating system programs are copyrightable because operating systems are instructions as opposed to processes and are expressions of the idea of the function rather than "purely utilitarian works."[111]

1994); Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992); Whelan Assocs., Inc. v. Jaslow Dental Lab., 797 F.2d 1222, 1234 (3d Cir. 1986); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 60 (D. Mass. 1990).

104. 714 F.2d 1240 (3d Cir. 1983).

105. *See id.* at 1250-53.

106. *See id.* at 1243.

107. *See id.* at 1243-45.

108. *Id.* at 1249.

109. *See id.* at 1243.

> The ROM (Read Only Memory) is an internal permanent memory device consisting of a semi-conductor "chip" which is incorporated into the circuitry of the computer. A program in object code is embedded on a ROM before it is incorporated in the computer. Information stored on a ROM can only be read, not erased or rewritten.

Id.

110. *See id.* at 1253.

111. *Id.* at 1251-53.

In the reasoning for its decision, the court noted that "[a]lthough section 102(a) does not expressly list computer programs as works of authorship, the legislative history suggests that programs were considered copyrightable as literary works."[112] Congress had created a Commission on New Technological Uses (CONTU) to "study, *inter alia*, computer uses of copyrighted works[,]" and had thus, based on CONTU's recommendation, amended 17 U.S.C. § 101 to include a definition of a computer program and had added new § 117, which limits exclusive copyright rights for computer programs.[113] The court stated that the new and amended provisions "clearly indicate[d] that programs are copyrightable and are otherwise afforded copyright protection."[114] The court concluded that programs are "literary works" because "[t]he definition of 'literary works' in section 101 includes expression not only in words but also 'numbers, or other . . . numerical symbols or indicia', thereby expanding the common usage of 'literary works.' "[115] Courts often cite *Apple* for the proposition that literal elements of software are undoubtedly protected from copying.[116]

3. Broad Protection for Software: Whelan Associates, Inc. v. Jaslow Dental Laboratories, Inc.

After *Apple v. Franklin*, the Third Circuit in *Whelan Associates, Inc. v. Jaslow Dental Laboratories, Inc.*[117] continued to use traditional copyright doctrines, including idea/expression merger, *scenes a faire*, and public domain, to determine substantial similarity without the benefit of the AFC test pattern.[118] However, the most significant addition to the development of the still-to-be-formalized AFC test in *Whelan* was the Third Circuit's holding that a computer program's "structure, sequence, and organization" may be protected.[119] The court analyzed the nonliteral elements of the program at issue by considering whether the "purpose or function of a utilitarian

112. *Id.* at 1247 (citing H.R. REP. NO. 94-1476, at 54 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667).
113. *Id.* at 1247-48.
114. *Id.* at 1248.
115. *Id.* at 1249.
116. *See* Barry, *supra* note 5, at 1319 & n.61.
117. 797 F.2d 1222 (3d Cir. 1986).
118. *See id.* at 1233-42.
119. *Id.* at 1248.

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."[120]

In *Whelan*, the developer of a computer program made for a single dental laboratory sued another dental laboratory for re-creating the program in another computer language but with a similar purpose and for then distributing the program.[121] Acknowledging the lack of literal similarity between the two programs, the plaintiff alleged defendant's program infringed the overall structure of its program.[122] The Court stated that copyrights of computer programs may be infringed when the nonliteral elements are copied, analogizing them to literary works.[123]

Commentators often criticize *Whelan* for its broad protection of software because the court held that the means to achieve the main purpose of the software served as the software's expression.[124] In effect, "[t]he court was willing to protect everything that was viewed as not being inherently part of the broad 'purpose' or underlying 'idea' that a program sought to effectuate."[125] One commentator has stated that the court "seriously distort[ed] copyright law, especially with respect to the fundamental idea/expression dichotomy."[126] Further, the *Whelan* court failed to "recognize that different ideas may underlie the different levels of the program"[127] and that a program may contain many ideas other than its main purpose.[128] The understanding that different ideas exist in a

120. *Id.* at 1236 (emphasis omitted).
121. *See id.* at 1224-27.
122. *See id.* at 1233.
123. *See id.* at 1234. The court cited several literary works infringement cases in which infringement occurred even though the literal elements were not substantially similar. *See id.* The cases cited included the following: Twentieth Century-Fox Film Corp. v. MCA, Inc., 715 F.2d 1327 (9th Cir. 1983); Sid & Marty Krofft Television Prod., Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977); Roth Greeting Cards v. United Card Co., 429 F.2d 1106 (9th Cir. 1970); and Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930). *See id.*
124. *See* Barry, *supra* note 5, at 1321 & n.77.
125. Patricia A. Martone, *Intellectual Property Protection for Computer Software—As Copyright Protection Narrows, Can Patents Fill the Gap?*, 479 PRACTISING L. INST./ PATENTS, COPYRIGHTS 599, 602 (1997).
126. Amin, *supra* note 2, at 35.
127. *Id.* at 35 n.116 (quoting Marc T. Kretschmer, Note, *Copyright Protection for Software Architecture: Just Say No!*, 3 COLUM. BUS. L. REV. 823, 839 (1988)).
128. *See id.*

software program takes form in discerning different levels of abstraction in the AFC test.

4. *A Similar Test:* Lotus Development Corp. v. Paperback Software International

The district court in *Lotus Development Corp. v. Paperback Software International*[129] set out a three-part "legal test" for software copyrightability that is somewhat similar to the AFC test.[130] Paperback argued that it should be allowed to copy Lotus 1-2-3's arrangement, names of commands, and menus so that Paperback's spreadsheet program would be compatible with Lotus's program.[131] However, the district court held that the Lotus 1-2-3 user interface, its menu command structure, was copyrightable.[132]

Paperback is important for two reasons. First, the court stated that the Copyright Act "does not bar copyrightability merely because the originality of the expression becomes associated, in the marketplace, with usefulness of the work to a degree and in dimensions not previously achieved by other products on the market."[133] Therefore, if other courts follow *Paperback*, certain software would retain protection even when the software gains so much of the market share that it becomes a de facto standard.[134]

Second, the court set out a three-part "legal test" for copyrightability.[135] Under the first step, a court may "focus upon alternatives that counsel may suggest, or the court may conceive, *along the scale from the most generalized conception to the most particularized*, and choose some formulation—some conception or definition of the 'idea'—for the purpose of distinguishing between the idea and its expression."[136] The second step focuses on the idea/expression merger by determining "whether an alleged expression of the idea is limited to elements

---

129. 740 F. Supp. 37 (D. Mass. 1990).
130. *See id.* at 60-61.
131. *See id.* at 68-69.
132. *See id.* at 68.
133. *Id.* at 58.
134. *See* Barry, *supra* note 5, at 1343-44.
135. *See Paperback*, 740 F. Supp. at 60-61.
136. *Id.* at 60 (emphasis in original). The first step involves analyzing nonliteral elements to determine the idea and the expression. *See* Martone, *supra* note 125, at 603.

essential to expression of *that* idea . . . or instead includes identifiable elements of expression not essential to every expression of that idea."[137] The third step focuses on whether identified elements of expression, separate from the idea, constitute a "substantial part of the allegedly copyrightable 'work.' "[138]

The *Paperback* legal test is somewhat like the AFC testing method, but it does not include the detail of abstraction and filtration within its application of traditional copyright doctrine.[139] However, *Lotus Development Corp. v. Borland International, Inc.*,[140] which concerned the same facts and plaintiff as in *Paperback*, effectively overruled the *Paperback* decision.[141] Before discussing the *Borland* decision, the adoption of the AFC test in *Computer Associates International v. Altai, Inc.*[142] should be considered.

5. *Firm Adoption of AFC Testing:* Computer Associates International v. Altai, Inc.

In *Altai*, the Second Circuit adopted the AFC test for determining whether nonliteral elements of computer software have been infringed.[143] There, the defendant copied thirty percent of plaintiff's code to develop a computer program very similar to plaintiff's program, which translated language of other programs into language that a particular computer could understand.[144] In other words, the operating system provided compatibility to other programs.[145]

The court expressed dissatisfaction with the methods adopted by other courts in deciding whether and to what extent nonliteral aspects of programs were copyrightable.[146] The court recognized

---

137. *Id.* at 61 (emphasis in original).
138. *Id.* Some commentators believe that *Paperback* is also significant because it rejects the useful article concept as a basis for denying copyright protection. *See* Barry, *supra* note 5, at 1329.
139. *See* discussion, *infra* Part II.A.5.
140. 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court*, 116 S. Ct. 804 (1996).
141. *See infra* notes 195-208 and accompanying text; *see also* Barry, *supra* note 5, at 1329 n.132.
142. 982 F.2d 693 (2d Cir. 1992).
143. *See id.* at 706.
144. *See id.* at 699-700.
145. *See id.*
146. *See id.* at 696.

that computers are utilitarian in nature and therefore relied on *Baker v. Selden*[147] to conclude that "those elements of a computer program that are necessarily incidental to its function are similarly unprotectable."[148]

The court criticized the *Whelan* court for failing to realize that more than one idea may underlie a computer program and for using the terms "structure," "sequence," and "organization" synonymously, which showed that the *Whelan* court did not understand a program's method of operation.[149]

In a passage other courts now often quote, the *Altai* court set forth the AFC test for determining software copyrightability:

> In ascertaining substantial similarity under this approach, a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernal, or possible kernals, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of the allegedly infringing program. The result of this comparison will determine whether the protectable elements of the programs at issue are substantially similar so as to warrant a finding of infringement.[150]

To apply the AFC test, a court must first engage in abstraction.[151] Abstraction basically retraces the program designer's steps in the opposite order of creation.[152]

> As an anatomical guide to this procedure, the following description is helpful: At the lowest level of abstraction, a computer program may be thought of in its entirety as a set of individual instructions organized into a hierarchy of modules. At a higher level of abstraction, the instructions in the lowest-level modules may be replaced conceptually by the

147. 101 U.S. 99 (1879).
148. *Altai*, 982 F.2d at 705.
149. *Id.* at 705-06.
150. *Id.* at 706.
151. *See id.* at 707.
152. *See id.* "[T]he process of abstraction is the opposite of the process that a programmer would use to create the program." Carleton, *supra* note 97, at 929. The programmer starts with the general idea and ends by writing the code. *See id.*

> functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace the implementations of those modules in terms of lower-level modules and instructions, until finally, one is left with nothing but the ultimate function of the program .... At low levels of abstraction, a program's structure may be quite complex; at the highest level it is trivial.[153]

Beginning with the code and ending with the program's "ultimate function," the court dissects the copied program's structure and isolates each level of abstraction within the structure.[154] A programmer works in steps from the abstract to the particular with each successive step being more specific than the previous step.[155] The process of creating a program "produces natural levels of abstraction that the court can use in its inquiry."[156]

Next, the court filters out all nonprotectable expression by examining the program's structural components at the previously identified levels of abstraction by using traditional doctrines of copyright law.[157] "Filtration is the key to achieving narrow copyright protection."[158]

The court begins the process of filtration by looking at the elements dictated by efficiency.[159] Filtration analyzes idea/expression merger.[160] The *Altai* court discussed the difficulty of applying the merger doctrine to computer programs.[161] When only a limited number of ways exist to express an idea and that expression is the only essential means of making a result happen, "efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options."[162] The court then must inquire whether, in the specific area of computer programs, the use of a particular structure is necessary in an efficiency context to implement a process; if the

153. *Altai*, 982 F.2d at 707 (quoting Steven R. Englund, Note, *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)).
154. *Id.*
155. *See* Carleton, *supra* note 97, at 928.
156. *Id.* at 925.
157. *See Altai*, 982 F.2d at 707.
158. Barry, *supra* note 5, at 1345.
159. *See Altai*, 982 F.2d at 707.
160. *See id.* at 707-08.
161. *See id.* at 708.
162. *Id.*

answer is yes, the expression has merged with the idea and becomes uncopyrightable.[163]

The *Altai* court next discussed elements dictated by external factors within the filtration step.[164] At this point, the court considered whether the elements of the program come under the exception of the *scenes a faire* doctrine.[165] In the computer context, the *scenes a faire* doctrine means that a software design company may not be able to design a program to accomplish certain tasks without using standard computing techniques, including compatibility requirements, design standards, and industry demands.[166]

A court must also consider whether any of the elements are taken from the public domain and then must filter out those elements.[167] The *Altai* court saw "no reason to make an exception to this rule for elements of a computer program that have entered the public domain by virtue of freely accessible program exchanges and the like."[168]

A court next determines whether a substantial similarity exists between the plaintiff's and the defendant's expression.[169] The court has already filtered out all nonprotectable expression, and in this last step, it can concentrate on whether the defendant copied any aspect of the plaintiff's protectable expression.[170] "*De minimis* copying of small code fragments, the inclusion in the copying program of similar functions, or similar behavior between the two programs is not substantial . . . if the *amount* taken is minimal or insignificant in relation to the total quantity that is the whole work."[171] Qualitative and quantitative considerations for substantial similarity must be considered in the final comparison step.[172]

The *Altai* court disagreed with the policy argument in *Paperback* that incentives for creation may give broad copyright protection, by stating that such incentive-based arguments have

---

163. *See id.*
164. *See id.* at 709.
165. *See id.*
166. *See id.* at 709-10.
167. *See id.* at 710.
168. *Id.*
169. *See id.*
170. *Id.*
171. Carleton, *supra* note 97, at 933.
172. *See id.* at 933-34; *supra* Part I.D. (discussing substantial similarity in proving copyright infringement).

a "corrosive effect on certain fundamental tenets of copyright doctrine."[173] The court expected the AFC test to narrow the scope of protection for computer software.[174]

6. *A Different Test:* Brown Bag Software v. Symantec Corp.

In contrast, the Ninth Circuit in *Brown Bag Software v. Symantec Corp.*[175] created a different test with only two steps.[176] There, Symantec developed the first successful outlining program, "ThinkTank."[177] However, another computer company, Softworks Development, sold another outlining program, "PC-Outline," to Brown Bag that was similar to ThinkTank.[178] Softworks Development thereafter developed and sold an outlining program, "Grandview," to Symantec.[179] Brown Bag sued Symantec, alleging that Grandview infringed Brown Bag's PC-Outline copyright.[180]

The Ninth Circuit found in favor of Symantec under its two-step test for infringement.[181] Under the first step, called the "objective/extrinsic" step, the court analytically dissected the copyrighted work through various copyright doctrines and analyzed the protected elements for substantial similarity of expression and idea.[182] This portion of the test involves an objective analysis of expression.[183] If substantial similarity exists in the first step, then under the second step, called the "subjective/intrinsic" step, an ordinary, reasonable person standard is used to compare the works, including elements eliminated from the extrinsic test, for substantial similarity.[184]

---

173. *Altai*, 982 F.2d at 712.
174. *See id.*
175. 960 F.2d 1465 (9th Cir. 1992).
176. *See id.* at 1475.
177. *See id.* at 1468.
178. *See id.*
179. *See id.*
180. *See id.* at 1469.
181. *See id.* at 1478.
182. *See* Barry, *supra* note 5, at 1330.
183. *See Brown Bag*, 960 F.2d at 1475.
184. *See id.*; Barry, *supra* note 5, at 1330.

7. *Adoption of the AFC Test in the Tenth Circuit:* Gates Rubber Co. v. Bando Chemicals Industries, Ltd.

Unlike the Ninth Circuit in *Brown Bag*, the Tenth Circuit in *Gates Rubber Co. v. Bando Chemicals Industries, Ltd.*[185] adopted the AFC test from *Altai*, but added more detail to the analysis.[186] In *Gates*, an industrial machinery rubber belt manufacturer developed a computer program to aid in the efficient and accurate selection of belts.[187] Bando, Gates's competitor, developed a computer program similar to Gates's program, and Gates sued for copyright infringement.[188]

In determining whether Bando's software infringed, the *Gates* court adopted the AFC test and added more detail and guidance in the analysis.[189] For the abstraction step, the court divided the program into "at least six levels of generally declining abstraction: (i) the main purpose, (ii) the program structure or architecture, (iii) modules, (iv) algorithms and data structures, (v) source code, and (vi) object code."[190]

In the filtration step, the *Gates* court utilized not only the idea/expression dichotomy spelled out in *Altai*, but also detailed the process-expression dichotomy, which addressed the copyrightability of utilitarian processes as in *Baker v. Selden*.[191] "Although processes themselves are not copyrightable, an author's description of that process, so long as it incorporates

185. 9 F.3d 823 (10th Cir. 1993).
186. *See id.* at 834, 836-39.
187. *See id.* at 830.
188. *See id.* at 831.
189. *See id.* at 834-41.
190. *Id.* at 835. The court specifically described the levels of abstraction. *See id.* The main program's purpose must be described specifically without technical language. *See id.* The program's architecture or structure describes how the program "operates in terms of its various functions, which are performed by discrete modules, and how each of these modules interact [sic] with each other." *See id.* A module consists of operations and data types. *See id.* Algorithms and data structures are more specific. *See id.* Algorithms are "specific series of steps that accomplish a particular operation." *Id.* Data structures are specific representatives of data types consisting of variables, values, and like elements. *See id.* "Source code is the literal text of a program's instructions written in a particular programming language." *Id.* "Object code is the literal text of a computer program written in a binary language through which the computer directly receives its instructions." *Id.* Although the *Altai* court recognized levels of abstraction, it did not specifically describe them. *See supra* notes 151-56 and accompanying text.
191. *Gates*, 9 F.3d at 836-37.

some originality, may be protectable."[192] In addition, the court filtered out facts since facts themselves are not copyrightable, but the court recognized that the arrangement or selection of facts may be protectable as a compilation.[193] Therefore, *Gates* accepted *Altai*'s AFC test and only elaborated on the steps.[194]

8. *Rejection of the AFC Test for User Interfaces:* Lotus Development Corp. v. Borland International, Inc.

The First Circuit rejected the AFC test for determining the copyrightability of a menu command hierarchy.[195] Four days after a Massachusetts federal district court decided *Paperback*, Lotus sued Borland in *Lotus Development Corp. v. Borland International, Inc.*[196] The facts in *Borland* were essentially the same as the facts in *Paperback*: Borland's Quattro and Quattro Pro spreadsheet programs included "a virtually identical copy of the entire 1-2-3 menu tree"[197] of the Lotus spreadsheet program.[198] The First Circuit's decision reversed the series of *Borland* decisions,[199] and also effectively overruled the district court's opinion in *Paperback*.[200] "Computer software is sold on a national basis, and consistent rules across circuits are needed. Recognizing the problem, the United States Supreme Court granted certiorari, as it often does when faced with a split among the circuits. Unfortunately, the Court's 4-4 vote left the circuit split intact."[201]

The First Circuit rejected the AFC test developed by the *Altai* court when it concluded that "[w]hile the *Altai* test may provide a

192. *Id.* at 837.
193. *See id.* Compilations are discussed in further detail *infra* Part V.C.
194. *See id.* at 841.
195. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided court*, 116 S. Ct. 804 (1996).
196. *See id.* at 810; Comment, Lotus v. Borland*: Is Anything Left in Software to Copyright?*, 41 WAYNE L. REV. 1713, 1715 (1995).
197. *Borland*, 49 F.3d at 810 (quoting Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202, 212 (D. Mass 1993) (emphasis omitted)).
198. *See id.*
199. *See* Comment, *supra* note 196, at 1715 & n.15 (citing Lotus Dev. Corp. v. Borland Int'l, Inc., 788 F. Supp. 78 (D. Mass. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203 (D. Mass. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202 (D. Mass 1993); Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 223 (D. Mass. 1993)).
200. *See id.* at 1715.
201. *Id.* at 1739.

useful framework for assessing the alleged nonliteral copying of computer code, we find it to be of little help in assessing whether the literal copying of a menu command hierarchy constitutes copyright infringement."[202] Additionally, the First Circuit criticized the *Altai* test for obscuring through its abstraction and filtration steps the "more fundamental question of whether a menu command hierarchy can be copyrighted at all."[203]

However, the First Circuit failed to create its own test and merely held that the Lotus menu command hierarchy was an uncopyrightable "method of operation."[204] Because the court found that methods of operation include any "means by which a user operates something,"[205] "virtually every nonliteral element of a computer program can be considered to be a 'method of operation.' "[206]

One commentator has stated that "[d]ismissing the copyrightability of the Lotus menu command hierarchy because it is a 'method of operation' represents a step back in the evolution of copyright law as applied to computer programs."[207] Another commentator has said that the *Borland* decision could result in "greater technological innovation or a relative stagnation caused by the belief that copyright law is insufficient for the protection of software."[208] Lacking guidance from the Supreme Court, the circuits continued to create or adopt different methods for determining the copyrightability of computer software.

9. *Adoption of the AFC Test for User Interfaces:* Engineering Dynamics, Inc. v. Structural Software

In contrast to the First Circuit in *Borland*, the Fifth Circuit in *Engineering Dynamics, Inc. v. Structural Software*[209] adopted the AFC test for determining whether computer user interfaces, input formats, and output reports were copyrightable.[210]

---

202. *Borland*, 49 F.3d at 815.
203. *Id.*
204. *Id.*
205. Martone, *supra* note 125, at 607-08.
206. *Id.* at 608.
207. Porter, *supra* note 17, at 307.
208. Jason A. Whong & Andrew T.S. Lee, Casenote, Lotus v. Borland*: Defining the Limits of Software Copyright Protection*, 12 COMPUTER & HIGH TECH. L.J. 207, 217 (1996).
209. 26 F.3d 1335 (5th Cir. 1994).
210. *See id.* at 1343; *see also* Skon, *supra* note 26, at 1075-80 (discussing why this

Engineering Dynamics popularized a computer program "to solve engineering problems in the field of structural analysis."[211] However, Structural Software marketed its own program that "utilized precisely the same input formats and input sequence" as the Engineering Dynamics program.[212] Engineering Dynamics alleged copyright infringement for fifty-six input formats, as well as additional output report formats and user manuals.[213] The Fifth Circuit reversed the district court's holding that input formats and output formats were uncopyrightable and remanded to determine whether structural software infringed Engineering Dynamics' copyright.[214]

10. A Different Test: Apple Computer, Inc. v. Microsoft Corp.

The Ninth Circuit in *Apple Computer, Inc. v. Microsoft Corp.*[215] set forth a different test from the AFC test to determine copyright protection.[216] In the Ninth Circuit, a plaintiff initially must identify the source of the alleged similarity between plaintiff's and defendant's works.[217] Then, using analytic dissection with traditional copyright principles, the court determines whether the similarities are subject to copyright protection.[218] After dissection, the court must define the scope of the plaintiff's copyright by determining whether the work will receive broad or thin copyright protection.[219] In this "virtual identity" test, only thin protection is given for a program's overall "look and feel," rather than a broader-based protection against

---

commentator thinks that the court in *Engineering Dynamics* misapplied the *Altai* test).

211. *Engineering Dynamics*, 26 F.3d at 1338. The program was originally designed by Synercom although Engineering Dynamics added developments of its own. *See id.*

212. *Id.* The quoted language was used by the court to refer to Engineering Dynamics's use of Synercom's formats, but it is equally applicable to Structural Software's use of EDI's formats. *See id.*

213. *See id.* at 1339.

214. *See id.* at 1351.

215. 35 F.3d 1435 (9th Cir. 1994).

216. *See id.* at 1443.

217. *See id.*

218. *See id.*

219. *See id.*

substantial similarity.[220] The court stated that other courts perform the same analysis yet describe it differently.[221]

In the *Apple* case, Apple sued Microsoft for copyright infringement of the Macintosh's graphical user interface after Microsoft released Microsoft Windows 1.0.[222] The district court found that "almost all the similarities [sprang] . . . from basic ideas and their obvious expression, . . .[and] illicit copying could [have occurred] only if the works as a whole [were] virtually identical."[223] One commentator has said that after the decision in *Apple*, if a claim is based on the "look and feel" of individual unprotected elements of expression, the given protection is thin and infringement will be found only if the expression is "virtually identical."[224]

B. The Split in the Circuits

The circuits remain split on what test should be applied to determine software copyrightability and infringement, which causes much confusion within each circuit. "While not a direct split, there is an effective split between the First Circuit and the Second, Tenth, and Federal Circuits."[225] The Second Circuit in *Altai* and the Tenth Circuit in *Gates* have adopted the AFC testing method. The Ninth Circuit in *Brown Bag* and *Apple* has adopted a similar method. The First Circuit in *Lotus v. Borland* has criticized the *Altai* decision and refused to apply the AFC test to user interfaces. However, the Fifth Circuit in *Engineering Dynamics* applied the AFC test to user interfaces.

---

220. Martone, *supra* note 125, at 606-07. There can be no infringement of a compilation unless the works are virtually identical. *See Apple*, 35 F.3d at 1447. The 11th Circuit adopted the "virtual identity" test for analyzing claims of compilation copyright infringement in Mitek Holdings, Inc. v. Arce Eng. Co., 89 F.3d 1548, 1558 (11th Cir. 1996).

221. *Apple*, 35 F.3d at 1445 (listing *inter alia, Computer Assocs. Int'l v. Altai, Gates Rubber Co. v. Bando Chem. Indus., Ltd., Engineering Dynamics, Inc. v. Structural Software*, and *Lotus Dev. Corp. v. Paperback Software Int'l*).

222. *See* Rodger R. Cole, *Substantial Similarity in the Ninth Circuit: A "Virtually Identical" "Look and Feel"?*, 11 SANTA CLARA COMPUTER & HIGH TECH. L.J. 417, 417-18 (1995).

223. *Apple*, 35 F.3d at 1447.

224. Cole, *supra* note 222, at 417.

225. Barry, *supra* note 5, at 1346 n.240.

III. *BATEMAN V. MNEMONICS, INC.*

In *Bateman v. Mnemonics, Inc.*,[226] the Eleventh Circuit addressed two issues of first impression in that circuit.[227] The court decided that the AFC testing method should be used to analyze computer software copyright infringement claims and held that interface specifications may be entitled to copyright protection, depending upon the facts of the case at hand.[228]

A. Background

The facts of the *Bateman* case are complex. The plaintiffs in this action, Brian E. Bateman and Charles H. Fricker, filed suit against Mnemonics, Inc., Harry Thompson, David Katz, Parking Automation Corp. (PAC), BCS, Inc. (BCS), and Robert Brunet.[229] Bateman and Fricker are engineers who authored hardware logic diagrams for a single board computer (SBC) created for use in automated parking systems.[230] Bateman alone developed software for a general purpose SBC operating system (SBCOS) for use in parking systems, cash drawer management, and energy management.[231] Bateman and Fricker registered a copyright for both Bateman's software and Bateman and Fricker's hardware logic diagrams.[232]

In the beginning, Bateman worked with BCS and also with its principal, Mr. Brunet, who manufactured and sold computer boards to Generex Corporation among others.[233] Generex, a parking company, wanted to buy a computer board and operating system for use in its business.[234] In response to Generex's request, Bateman and Fricker assisted Brunet in designing a SBC containing Bateman's SBCOS software.[235] BCS sold many of the SBC boards to Generex, and thereafter Generex created an application program to interoperate with the SBCOS.[236]

---

226. 79 F.3d 1532 (11th Cir. 1996).
227. *See id.* at 1536.
228. *See id.*
229. *See id.*
230. *See id.*
231. *See id.* at 1536 n.3.
232. *See id.* at 1536.
233. *See id.*
234. *See id.* at 1537.
235. *See id.*
236. *See id.* "The SBCOS comes in chip form and was one of the components that was integrated into the circuit board, or SBC, to render it operational." *Id.* at 1537

PAC, a wholly owned subsidiary of Mnemonics, Inc., acquired Generex, and BCS began selling its SBCs to PAC.[237] Bateman and PAC decided to do business directly with each other.[238] Bateman and Fricker agreed to design a second single board computer (SBC2) with an updated version of the SBCOS software, including programmable array logic (PAL) technology.[239] Bateman and Fricker thereafter delivered the SBC2 and the engineering for the board to PAC for manufacture.[240] Bateman then told BCS to stop using his SBCOS software in the SBCs being sold.[241]

The business relationship between PAC and Bateman/Fricker began to deteriorate when PAC started having field problems with the SBC2s.[242] PAC requested the source code for the SBCOS and "rights to the SBC2 circuit design in exchange for a release of claims based on the inoperability of the delivered SBC2s . . . ."[243] Bateman and Fricker refused and terminated the contract with PAC.[244]

PAC disassembled and decompiled part of the SBCOS from a computer chip on one of the SBC2s in order to create its own operating system to interoperate with the program that was written to be compatible with the SBCOS.[245] PAC claimed that this reverse engineering was necessary because Bateman had failed to deliver the source code for the SBCOS.[246] Moreover, PAC claimed that the "elements of the program that were necessary for compatibility with the preexisting application

---

n.7.

237. *See id.* at 1537.
238. *See id.* at 1538.
239. *See id.*
240. *See id.*
241. *See id.*
242. *See id.*
243. *Id.* at 1539. PAC paid Bateman and Fricker $35,000 for the SBC2s. *See id.*
244. *See id.*
245. *See id.*

> PAC claims that such reverse engineering was necessary to develop an operating system compatible with its application program, which in turn was written to be compatible with Bateman's SBCOS. Although the issue is not directly before us on appeal, it will likely appear on remand, and thus we think it proper to address it and thereby provide some guidance to the district court on remand.

Id. at 1539 n.18.

246. *See id.* at 1539-40.

program"[247] were the only parts incorporated into PAC's new operating system, Lane Control Computer Operating System (LCCOS).[248] In addition, PAC developed a new circuit board to replace the SBC2.[249]

After termination of the contract with PAC, Bateman and Fricker filed suit in Florida state court for breach of contract.[250] Once Bateman and Fricker learned that PAC was selling a computer circuit board that "appeared to perform the same functions as their SBC2,"[251] they moved their case to federal court and alleged several additional claims, including copyright infringement in the SBCOS, the hardware logic diagrams for the computer circuit board, and the PAL software; false designation of origin; common law unfair competition; and theft of trade secrets.[252] The jury found copyright infringement in the SBCOS software and theft of trade secrets and awarded $525,000 to Bateman and Fricker.[253] PAC appealed the judgment.[254]

B. Issues Appealed

PAC raised eight issues on appeal, and Bateman and Fricker raised one issue in their cross-appeal.[255] For purposes of this Comment, the discussion of issues appealed will be limited to whether PAC's allegedly infringing program was substantially similar to protectable elements of Bateman and Fricker's SBCOS code and the SBC2 logic circuit diagrams, and whether PAC's copying of any original elements was a fair use.[256] Judge Birch

---

247. *Id.* at 1540.
248. *See id.*
249. *Id.*
250. *See id.*
251. *Id.*
252. *See id.* at 1536, 1540.
253. *See id.* at 1536. PAC conceded that Bateman and Fricker were entitled to an injunction on the false designation of origin and common law unfair competition claims, and no appeal was taken. *See id.* at 1536 n.2.
254. *See id.* at 1536. Bateman and Fricker cross-appealed, "asserting that the district court abused its discretion in refusing to award exemplary damages to them on the trade secret claim." *Id.* Analysis of the trade secret claim is beyond the scope of this Comment.
255. *See id.* at 1540.
256. *See id.* (listing complete issues that were appealed). PAC did not contest Bateman and Fricker's ownership of a valid copyright, and PAC admitted that it did copy portions of the SBCOS code and the SBC2 logic circuit diagrams. *See id.* at 1542. However, Bateman and Fricker still had to prove the two works were substantially similar, and they had to challenge successfully "the putative infringer's

of the Eleventh Circuit stated that "[a]t the heart of these issues is the parties' dispute over the district court's jury instructions on literal and nonliteral copying . . . ."[257]

C. *The District Court's Jury Instructions*

After listing the issues on appeal, the court noted that a deferential standard of review applied to the district court's instructions, as long as those jury instructions were correct in the law.[258] Further, the instructions as a whole must have sufficiently and correctly instructed the jury so that the jury understood the issues and was not misled.[259] The court stressed "the need for proper guidance"[260] in instructing the jury in the area of copyright law, which can be complex and foreign to jurors.[261]

First, the court attacked the district court's instruction that limited the filtration step of the AFC test to nonliteral similarity.[262] The district court used verbatim the instruction proposed by Bateman and Fricker for determining substantial similarity.[263] "PAC submitted its own version of a filtration

---

evidentiary assertion that the use made of the original portion of the copyrighted work is a fair use under 17 U.S.C. § 107 . . . ." *Id.*

257. *Id.* at 1542-43.

258. *See id.* at 1543 (citing United States v. Starke, 62 F.3d 1374, 1380 (11th Cir. 1995) (citing McElroy v. Firestone Tire & Rubber Co., 894 F.2d 1504, 1509 (11th Cir. 1990))).

259. *See id.* (quoting Wilkinson v. Carnival Cruise Lines, Inc., 920 F.2d 1560, 1569 (11th Cir. 1991)).

260. *Id.*

261. *See id.*

262. *See id.* at 1544.

263. *See id.*

The district court's instruction limited the filtration analysis to nonliteral similarity, stating that:

> Under the Nimmer test, substantial similarity of the non-literal elements is determined by comparing with the defendants' program, that protectable expression of the copyrighted work which remains after filtering out any portion of the copyrighted work, which represents only ideas, elements[ ] dictated solely by logic and efficiency, elements dictated by hardware or software standards, computer industry programming and practices or elements which are taken from the public domain.
>
> Even a qualitatively [sic] small amount of copied material which remains unfiltered may be sufficiently important to the operation of Mr. Bateman's operating system program to justify a finding of substantial similarity. For instance, a small portion of a structure or code of a program may nonetheless give it its

instruction, one that would direct the jury to filter out all nonprotectable expression, without regard to whether the similarity existed on the literal or nonliteral level."[264] PAC argued that merger, *scenes a faire*, and other traditional copyright originality challenges to literal copying should be applied in the filtration step, which is used to "separate protectable expression from nonprotectable material."[265] Over PAC's objections, the district court did not include literal similarities in its filtration instruction.[266]

The court recognized that the *Altai* AFC test was created by the Second Circuit to determine whether nonliteral elements of a computer program are protected by copyright.[267] The court also pointed out that "other circuits disagree on whether its application should be limited to instances of nonliteral copying, or whether it is equally applicable to cases involving literal copying."[268] The court discussed the *Lotus v. Borland* decision, which noted that the *Altai* test was not helpful in determining whether the "literal copying of a menu command hierarchy constitutes copyright infringement."[269] The court then contrasted *Lotus* with the *Gates* decision, which had adopted and added to the *Altai* test by stating that the test could be used to determine whether "menus and sorting criteria"[270] are protectable by copyright.[271]

The court saw the disagreement between the *Lotus* and *Gates* decisions as "more of a matter of semantics than substance."[272]

> distinctive features. In such a case, a finding of substantial similarity would be appropriate.

Id. (quoting trial record).

264. *Id.* at 1545 (emphasis omitted).

265. *Id.* at 1544. At the filtration step, the structural elements determined at the levels of abstraction are examined " 'to determine whether 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 . . . .' " *Id.* (quoting Computer Assocs. Int'l v. Altai, 982 F.2d 693, 707 (2d Cir. 1992)).

266. *See id.* at 1545.

267. *See id.*

268. *Id.*

269. *Id.* (citing Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995)).

270. *Id.* (quoting Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823 (10th Cir. 1993)).

271. *See id.*

272. *Id.*

> Even if the *Altai* test is limited to nonliteral copying, however, a parallel type of analysis must be undertaken in examining alleged instances of literal copying of computer code or screen displays. Whether one chooses to call the consideration of such generally recognized challenges to literal code copying as merger and efficiency "filtration" is of little consequence; what matters is that these well-established "defenses" are considered.[273]

The court pointed out that the *Gates* court "simply chose" to include the literal question in the filtration analysis.[274]

The court expressed no opinion as to whether literal copying questions should be considered in the filtration step or in a parallel analysis, but stated that some type of analysis is necessary.[275] The court noted in a footnote that a court must realize that different approaches adopted by other circuits are merely a means to the end of filtering out unprotectable expression, and that "parties become so engrossed in disputing what 'test' should apply that they lose sight of what the tests were designed to accomplish in the first place."[276]

The court held that the district court's jury instruction that filtration was limited to nonliteral copying was error as a "manifest distortion and misstatement of the law"[277] because the jury "must have concluded that any instances of literal copying of Bateman's code by PAC were by definition acts of copyright infringement."[278] Indeed, Bateman and Fricker presented substantial evidence of literal copying of computer code at trial.[279]

The court next attacked the district court's failure to give the jury instruction on the legal consequences of copying elements of the code based on compatibility and interoperability requirements.[280] The district court did give an instruction that stated computer programs were utilitarian articles that contain many elements that could be dictated by considerations of efficiency and external factors like compatibility and industry

---

273. *Id.*
274. *Id.*
275. *See id.*
276. *Id.* at 1545 n.27.
277. *Id.* at 1545.
278. *Id.*
279. *See id.*
280. *See id.* at 1546.

standard and demand.[281] Even though the district court's instruction was not "technically incorrect,"[282] it did not define compatibility and failed to instruct the jury on the legal consequences of finding that copying may have been dictated by compatibility requirements.[283] The court held that this instruction, considered in combination with the filtration instruction, failed to inform the jury that compatibility is a consideration applied at the literal level.[284]

The court addressed another matter of first impression, whether interface specifications are not entitled to copyright protection as a matter of law.[285] The court ruled that the view of interface specifications as uncopyrightable as a matter of law was "an incorrect statement of the law."[286]

The court then joined other circuits in holding that "external considerations such as compatibility may negate a finding of infringement."[287] In adopting this view, the court considered the *Altai* elaboration that programmers are often constrained to follow standard techniques, including the mechanical specifications of the computers on which the program will run and the compatibility requirements of programs with which the program is designed to operate.[288] The court also noted the *Sega* court's decision that many elements of a program are often dictated by compatibility requirements and industry demands.[289]

In holding that compatibility requirements may negate a claim of infringement, the court, in a footnote, advised that because such findings depend on the facts of each case, it would not devise a bright-line rule on the compatibility issue.[290] The court

281. *See id.*
282. *Id.*
283. *See id.*
284. *See id.*
285. *See id.* at 1547.
286. *Id.*
287. *Id.*
288. *See id.* (citing Computer Assocs. Int'l v. Altai, 982 F.2d 693, 709-10 (2d Cir. 1993)).
289. *See id.* (citing Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1992)); *see also* text accompanying *infra* note 300.
290. *See Bateman,* 79 F.3d at 1547 n.33. The court further stated that "[i]n no case, however, should copyright protection be extended to functional results obtained when program instructions are executed and such results are processes of the type better left to patent and trade secret protection." *Id.*

remanded the case for a new trial on whether the copyright in the SBCOS software and the hardware logic diagrams for the computer circuit board were infringed.[291]

IV. ANALYSIS OF *BATEMAN V. MNEMONICS*

The Eleventh Circuit, joining the Federal, Ninth, Fifth, and Second Circuits, found that "compatibility may negate a finding of infringement."[292] Although the lawfulness of disassembly of a computer program was not before the court on appeal, and therefore not an issue at trial, the Eleventh Circuit, in a footnote, provided the district court guidance on the issue because it would likely appear on remand.[293] After discussing the decisions of *Sega* and *Atari*, the court found the *Sega* opinion more persuasive.[294] "In other words, the Eleventh Circuit in effect directed the district court to find Mnemonic's disassembly for purposes of achieving interoperability to be a fair use."[295]

The court also explained the importance of separating idea from expression in computer programs in another footnote:[296]

> It is particularly important to exclude methods of operation and processes from the scope of copyright in computer programs because much of the contents of computer programs is patentable. Were we to permit an author to claim copyright protection for those elements of the work that should be the province of patent law, we would be undermining the competitive principles that are fundamental to the patent system.[297]

In the footnotes mentioned above, the court basically viewed "computer programs as utilitarian literary works which should

---

291. *See id.* at 1550. The Court reversed the judgment of the district court on the trade secret misappropriation allegation, instructing the district court to enter judgment as a matter of law for PAC on that count. *See id.*

292. Gurrola, *supra* note 86, at 1-2 (citing Engineering Dynamics, Inc., v. Structural Software, Inc., 26 F.3d 1335, 1340 (5th Cir. 1994); *Altai*, 982 F.2d at 706; *Sega*, 977 F.2d at 1527-28; Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 843 (Fed. Cir. 1992)).

293. *See Bateman*, 79 F.3d at 1539 n.18; Jonathan Band, *Interoperability After* Lotus v. Borland*: The Ball is in the Lower Courts Again*, 13 COMPUTER LAW. 11, 11 (1996).

294. *See* Band, *supra* note 293.

295. *Id.*

296. *See id.*

297. *Bateman*, 79 F.3d at 1541 n.21.

receive a 'thinner' scope of copyright protection than more expressive literary works such as novels and plays."[298]

The Eleventh Circuit also held that compatibility factors *may* deny copyright protection to elements of computer programs.[299] The court stated that "whether the protection is unavailable because these factors render the expression unoriginal, nonexpressive per 17 U.S.C. § 102(b), or whether these factors compel a finding of fair use, copyright estoppel, or misuse, the result is to deny copyright protection to portions of the computer program."[300] In another footnote, the court emphasized that a finding of compatibility to preclude protection depends on the facts of each case, so it would not form a bright line rule to address the issue.[301] "[G]iven the absence of any guidelines for when such compatibility needs may or may not result in unprotect-ability, it is difficult to know what to make of the decision."[302]

Although the court did not form a per se rule that "interface specifications can not [sic] receive copyright protection,"[303] the court gave five different theories under which interface specifications would be denied protection, including whether compatibility factors render the expression unoriginal or non-expressive and whether they compel the court to find fair use, copyright estoppel, or misuse.[304] The court "certainly penciled a line in"[305] when it ordered the district court to instruct the jury on "the legal consequences of copying dictated by compatibility requirements."[306]

> The *Bateman* decision does not define "interface specification," and also refers to "interface commands" and "operating system interface" in the same section. The Eleventh Circuit correctly recognized that it would be reckless for it to attach legal results to terms whose meaning and technological significance may change over time. Instead,

298. Band, *supra* note 293.
299. *See id.* at 12.
300. *Bateman*, 79 F.3d at 1547.
301. *See id.* at 1547 n.33; Band, *supra* note 293, at 12.
302. Mitchell Zimmerman, *Baystate Holding: Technical Interfaces Not Copyrightable—On to the First Circuit!*, ANDREWS COMPUTER & ONLINE INDUS. LITIG. REP. 24252, June 3, 1997.
303. Band, *supra* note 293, at 12.
304. *Id.*
305. *Id.*
306. *Id.*

it provided a framework within which elements related to interoperability should be examined.[307]

Of main importance in this Comment, the Eleventh Circuit also held that the *Altai* AFC test applies to instances of literal copying,[308] perhaps recognizing that "literal elements must be filtered too."[309] Mnemonics' case was "gutted" when the district court instructed the jury only to filter out nonliteral elements, which implied to the jury that literal elements are not filtered because they are always copyrightable.[310]

V. VIABILITY OF THE AFC TEST

Several views exist which state that the AFC testing method is not a viable test of computer software copyright infringement. A common complaint concerning the AFC testing method is the test's tendency to grant thin protection to programs. Others argue that a *sui generis* form of protection for software is needed to reward efficient programmers, the AFC test should protect compilations, and trade dress protection or patent protection should be applied to programs rather than copyright protection.

A. The AFC Test Tends to Provide Thin Protection

"As hardware, software, and interfaces in general become more compatible, more expression will be deemed unoriginal or indispensable, and there will be less copyrightable expression."[311] The sequence of applications of traditional copyright doctrines in the AFC testing method "tends to exclude elements of the software from the eventual comparison for substantial similarity, and so tends to result in relatively narrow copyright protection."[312] The broad protection granted to software under *Whelan* has been getting thinner with each subsequent case.[313]

---

307. *Id.*
308. Peter A. Wald & John J. Kirby, Jr., *Standards for Interoperability and the Copyright Protection of Computer Programs*, 449 PRACTISING L. INST./PATENTS, COPYRIGHTS 73, 97 (1996).
309. Joseph & Vogel, *supra* note 12, at 97.
310. *Id.*
311. Cole, *supra* note 222, at 428.
312. Barry, *supra* note 5, at 1343.
313. *See* David G. Luettgen, *Functional Usefulness vs. Communicative Usefulness: Thin Copyright Protection for the Nonliteral Elements of Computer Programs*, 4 TEX.

To cure problems with the AFC testing method, "a court must do more than insist on a rigorous abstraction-filtration-comparison analysis; it must then also properly apply this analytical framework"[314] and must notice the unique characteristics of functional works and use filtration to ensure that protection is not extended to ideas.[315] The lacking analysis in this area fails to give courts direction in applying the AFC test by neglecting to explain how and where to draw a line between idea and expression, also leaving the software industry with little guidance.[316] Further, courts may grant monopolies over ideas by overprotecting expression, thus preventing compatibility and stifling progress by preventing the building upon old ideas.[317]

B. A Sui Generis *Form of Protection is Needed to Reward Efficient Programmers*

Yet another view maintains that the abstraction test in *Altai* rewards "inefficient programmers over efficient ones,"[318] by eliminating from protection the efficient elements of programs, exemplifying how inappropriate copyright is for computer programs.[319] Federal courts do not consistently apply protection for software architecture, the structure and organization of the software.[320] The software architecture, which requires substantial time and investment to develop, is the true value of software.[321] Computer software needs a *sui generis* form of protection in order to provide the incentive to invest time and money on new software because "our country cannot afford to handicap its software industry by employing archaic laws that are ill-equipped to handle such high technology as computer software."[322] "[A]llowing rival manufacturers to copy each

---

INTELL. PROP. L.J. 233, 234 (1996).

314. McKinney, *supra* note 16, at 263.

315. *See id.*

316. *See id.*

317. *See id.*

318. Amin, *supra* note 2, at 37.

319. *See id.*

320. *See id.* at 34.

321. *See id.* at 35.

322. *Id.* at 40-41. Even "[t]he court in *Altai* acknowledged, '[W]e think that copyright registration—with its indiscriminating availability—is not ideally suited to deal with the highly dynamic technology of computer science.' " *Id.* at 37 (quoting Computer Assocs. Int'l v. Altai, 982 F.2d 693, 712 (2d Cir. 1993)).

others' products without consequence would discourage manufacturers from bringing those products to market."[323]

C. *The AFC Test Does Not Provide Protection for Compilations*

Another criticism maintains that the AFC test does not provide adequate protection for compilations. 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."[324] Software can be protected as a compilation if it is comprised solely of elements from the public domain, industry standards, and similar limiting doctrines.[325] Compilations receive thin copyright protection because the selection and arrangement is the only protectable element in fact-based works.[326] Compilation of unprotectable elements may be copyrightable if there is sufficient originality in the nonliteral selection and arrangement of the program.[327]

Dissecting software into parts before comparison "condemns the copyright to 'death by a thousand cuts,' "[328] so a program must be considered in its entirety.[329] Supporters of this view believe that the AFC test should incorporate compilation expression. AFC testing does not consider "selection and arrangement expression in program structure."[330] Although the AFC testing method has "sound doctrinal foundation" and provides a useful framework, it does not account for "creative authorship required to design program structure."[331] One commentator has written about the importance of compilation expression in object-oriented programming, which is predicted to be the issue in the next generation of software litigation.[332]

---

323. Comment, *supra* note 196, at 1739.
324. 17 U.S.C. § 101 (1996).
325. *See* Barry, *supra* note 5, at 1317.
326. *See* Wilkins, *supra* note 1, at 448; *see also supra* Part II.A.10 and accompanying text (discussing *Apple v. Microsoft*).
327. *See* Barry, *supra* note 5, at 1325.
328. Wilkins, *supra* note 1, at 443.
329. *See id.*
330. *Id.* at 435.
331. *Id.* at 436.
332. *See id.* "Object-oriented programmers create large programs out of preexisting software building blocks, an approach that offers many advantages but tends to limit possibilities for creative expression to the selection and arrangement of software 'objects.' " *Id.* "Object-oriented design is a new software paradigm that emphasizes the

AFC testing should be modified because in many cases, the selection and arrangement of a program will be the "only protectable element present in an object-oriented program."[333] Compilation principles should be applied in order to protect this design-level expression in computer software.[334] Dissection "diverts . . . attention away from the higher-level expression embodied in program structure and design."[335] *Altai*'s test "should incorporate compilation doctrine at each stage of [AFC testing]."[336]

D. Trade Dress Protection May Be Better to Protect Nonliteral Elements

Another view suggests trade dress protection may be better than copyright to protect the "look and feel" of programs because the scope of protection of nonliteral elements is "still unclear and has been narrowing in recent years."[337] Trade dress protection encompasses the "overall image of a product as presented to the public, including features such as size, shape, color, texture, graphics, or even sales techniques."[338]

To prove trade dress infringement, the owner must show that the trade dress is distinctive, that the alleged infringing trade dress creates a "likelihood of confusion as to source, affiliation, or sponsorship," and that the trade dress is nonfunctional.[339] Of main concern to computer software is the nonfunctional requirement for trade dress. "The test for functionality has been articulated in numerous ways, but the heart of the test appears to be whether the design affords utilitarian benefits that

---

use of self-contained, interchangeable software parts for constructing large systems." *Id.* at 444. "Most of the creative authorship in an object-oriented program exists at the level of designing the interaction of its constituent parts: selecting which software objects to use, arranging them into a functioning software whole, and coordinating their interaction." *Id.; see also* Michael Moon, *Survival in the Age of Telemedia*, NEW MEDIA, Oct. 7, 1996, at 20 (discussing future of object-oriented programming in a business context).

333. Wilkins, *supra* note 1, at 436.

334. *See id.*

335. *Id.* at 444.

336. *Id.* at 455; *see also id.* at 457-64 (suggesting ways to incorporate compilation doctrine during the AFC test).

337. Oratz, *supra* note 95, at 1.

338. *Id.* at 3. "Trade dress protection has also been applied to services as well as products, including the overall appearance of a restaurant chain." *Id.*

339. *Id.* at 3; *see also id.* (detailing requirements).

competitors cannot effectively duplicate through the use of alternative designs."[340] A combination or arrangement of functional elements as a whole may be allowed trade dress protection as long as the combination as a whole is not functional.[341] As seen with the useful article doctrine in denying copyright protection, it is questionable whether trade dress protection, with its functionality requirement akin to the useful article doctrine, would be available for elements of programs when copyright protection is not.[342] Because trade dress is a form of trademark protection, it may be registered with the Patent and Trademark Office (PTO).[343]

E. Patent Protection Should be Used for All Software

Cases such as *Baker v. Selden*, *Altai*, and *Bateman v. Mnemonics* have called upon patent laws for the protection of computer software.[344] As copyright protection has been narrowing for computer software, the "Federal Circuit and [PTO] have been expanding the availability of patent protection."[345] The courts and the PTO have grappled over the past twenty years with the argument that software is not patentable because programs consist of mathematical algorithms.[346] Historically, only tangible inventions were given patent protection.[347] However, "[t]he essence of software does not lie in a process, machine, manufactured article, or composition of matter, but rather in its underlying algorithm."[348] Because computer programs do not meet patent law's tangibility requirement, the physical process element was added to give programs protection.[349] In *Diamond v. Diehr*,[350] the Supreme Court held that when the algorithms in computer software are linked to a

340. *Id.* at 5; *see also id.* (describing other tests for functionality for trade dress protection).
341. *See id.* at 6.
342. *See generally id.*
343. *See id.* at 3.
344. *See* Martone, *supra* note 125, at 615-17.
345. *Id.* at 615.
346. *See id.; see also* Gottchalk v. Benson, 409 U.S. 63 (1972) (holding mathematical algorithms are not patentable).
347. *See* Martone, *supra* note 125, at 620.
348. *Id.*
349. *See id.* at 621.
350. 450 U.S. 175 (1981).

physical process, the software is patentable.[351] Recently, the courts have eased the tangibility requirement by allowing "computer programs embodied in a tangible medium, such as floppy diskettes," to be patentable.[352] In the PTO's *Examination Guidelines for Computer-Related Inventions*, effective March 29, 1996, the PTO adopts a "much more tolerant attitude towards software-based patent claims."[353] The new Guidelines now "recognize media (e.g. floppy disks) containing a copy of the computer program as an 'article of manufacture.' "[354] The new Guidelines also recognize the following: a "series of specific operational steps to be performed on or with the aid of a computer" is a process and therefore patentable; PTO examiners will focus on the whole invention, not just the mathematical algorithms, in determining the patentability of the computer program; the new three basic classes of nonstatutory subject matter are abstract ideas, laws of nature, and natural phenomena, meaning that "[a] claim containing a mathematical algorithm . . . is only nonstatutory if it represents, as a whole, an abstract idea rather than applied technology."[355] The result of this new leniency means that computer programs are now more easily patentable.

Therefore, some commentators argue that only patent protection should be available for computer software.[356] This argument stems from the belief that copyright is providing patent-like protection to computer programs.[357] "Copyright is designed to promote the dissemination of useful knowledge to humans. A computer program communicates only in the sense that a transistor inside the computer's microprocessor communicates to other transistors."[358] Copyright protects works that communicate under section 102(a), and the separability test of pictorial, graphic, or sculptural features shows that copyright is unwilling to protect anything other than communicative aspects of works.[359] Because patent law protects functional

351. *See* Martone, *supra* note 125, at 617.
352. *In re* Beauregard, 53 F.3d 1583, 1584 (Fed. Cir. 1995).
353. Martone, *supra* note 125, at 622.
354. *Id.*
355. *Id.* at 623-24.
356. *See* Luettgen, *supra* note 313, at 236.
357. *See id.* at 254.
358. *Id.* at 250-51.
359. *See id.* at 237, 241.

works, meaning "things which operate to achieve results,"[360] patent protection should be given to software.[361] "In theory, since computer programs are functional and copyright does not protect functionality, computer programs should not be protectable by copyright at all."[362]

CONCLUSION

Although the AFC test appears to be nearly a standard in the circuits for determining computer software copyrightability, confusion still exits among the circuits concerning which test is applicable and what elements of software, especially the nonliteral aspects, are protectable. The Eleventh Circuit in *Bateman* wisely adopted the AFC test and added more detail to determine if and when software will be protected. The *Bateman* court correctly deemed literal copying subject to the rigors of AFC testing and also accepted the fact that compatibility is a large factor in negating copyright infringement, as the computer is a functional work, which may be entitled only to thin protection.

Although the *Bateman* court correctly adopted the AFC test, subsequent revision and additions to the test will likely be necessary as technology advances. As state-of-the-art computer improvements, such as object-oriented programming, are being developed for market, the AFC test should incorporate selection, arrangement, and architecture expression to protect compilation programs, which are becoming extremely beneficial to the public. If competitors may copy programs with impunity, incentives to create and market innovative programs will decrease, thereby frustrating the goal of copyright law, which is to promote creative authorship for the public benefit.

With trade dress protection questionable and patent protection very expensive and more difficult to obtain than copyright, copyright protection and the AFC test will remain the most viable methods of protection for computer software. However, the AFC test should be modified as technology progresses.

Lisa M. Gable

---

360. *Id.* at 244.
361. *See id.*
362. *Id.* at 248.