Volume 6, Fall Issue, 1992

PRAGMATISM IN SOFTWARE COPYRIGHT: *COMPUTER ASSOCIATES V. ALTAI*

*John H. Butler**

INTRODUCTION

Over the last decade the growth of the computer software industry has spawned a significant amount of litigation concerning the extent to which the law prevents copying or imitating a program. Recent decisions display judicial disagreement about the nature and scope of this protection. In *Computer Associates International, Inc. v. Altai, Inc.*,[1] the Second Circuit attempted to clarify a confusing body of law and limit what it perceived to be the over-protection of computer software through copyright law. The Second Circuit's opinion, however, fails to establish a coherent doctrine and ultimately obfuscates the issues by down-playing the underlying tension in the copyright debate.

I. REVIEW OF COPYRIGHT LAW

The Copyright Act of 1976, as amended in 1980, grants copyright protection to computer programs.[2] Early judicial interpretations of the Copyright Act held that it prohibited literal copying of computer software.[3] Subsequent cases have grappled with the issue of how much protection courts should give to the non-literal elements of computer software.

The primary limit placed on this protection is the "idea-expression" dichotomy. Protection cannot extend "to any idea, procedure, process, system, method of operation, concept, principle, or discovery."[4] This fundamental limitation on the scope of copyright protection has been the source of much difficulty, for as Judge Learned Hand noted in *Nichols v.*

---

* J.D., Harvard Law School, Class of 1994

1. Nos. 90-7893 & 91-7935, 1992 U.S. App. LEXIS 14305 (2d Cir. June 22, 1992).

2. *See* Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 49 (D. Mass. 1990) (noting that computer programs fall within the definition of protected "literary works" set forth in 17 U.S.C. § 101 (1988)).

3. *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249-54 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984) (extending protection against literal copying to operating systems software).

4. 17 U.S.C. § 102(b) (1988).

Universal Pictures Corp., "[n]obody has ever been able to fix that boundary [between idea and expression], and nobody ever can."[5] The essence of the problem is easy to grasp: Because the ideas behind any computer program can be expressed at varying levels of generality, the decision of what to term "idea" and what to term "expression" is somewhat arbitrary.

An early case attempting to divide idea and expression in computer software was *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[6] After using Whelan's dental lab management program, Jaslow decided to create his own competing program. Although Jaslow's program was not a direct translation of Whelan's, both its structure and overall organization were substantially similar to the Whelan program.[7] The court relied on *Baker v. Seldon*[8] to support the rule that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."[9] Once the court determined that the purpose of Whelan's program was to "aid in the business operations of a dental laboratory" and that the structure of the program was not necessary to provide this function, the conclusion that the program's organization constituted protectable expression was "inescapable."[10]

More recently, in *Lotus Development Corp. v. Paperback Software International*,[11] a Massachusetts district court set forth what was, at that time, the most comprehensive discussion of this topic. Using an analytic approach somewhat similar to *Whelan*, the court found that the defendant's computer spreadsheet, "VP-Planner," infringed Lotus's copyright of the menu system on its spreadsheet program, "Lotus 1-2-3."[12]

---

5. 45 F.2d 119, 121 (2d Cir. 1930) (finding no copyright infringement of a play because only the play's general theme was copied).

6. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).

7. *Id.* at 1229.

8. 101 U.S. 99 (1879).

9. *Whelan*, 797 F.2d at 1236 (emphasis omitted).

10. *See id.* at 1238-39. *But see* Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1262 (5th Cir.), *cert. denied*, 484 U.S. 821 (1987) (denying preliminary injunction because allegedly infringing structural aspects of program were dictated by the cotton market); Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003, 1012-14 (N.D. Tex. 1978) (finding that a program's input formats were unprotectable because no expression could be separated from the idea). *Synercom* thus leans towards protection only against literal copying, although "it would probably be a violation to take a detailed description of a particular problem solution . . . and program [it]." *Id.* at 1013 n.5.

11. 740 F. Supp. 37 (D Mass. 1990).

12. *See id.* at 70.

The critical first step in the analysis was the decision that "with the Copyright Act of 1976 and the 1980 amendments to that Act, Congress manifested an intention to use the idea-expression distinction as part of the test of copyrightability for computer programs."[13] The court then utilized a three-part test to separate protectable expression from unprotected idea: choose the appropriate level of specificity or generality of the program's idea, screen out expression that is necessary to the expression of the idea, and decide if the remaining protectable elements are a substantial (not only quantitatively, but also qualitatively) part of the work.[14]

The *Lotus* court explicitly rejected three major arguments advanced by the defendants. First, the court refused to adopt a "function-expression" test that would bar copyright protection for non-literal aspects of computer programs that were functional.[15] Second, the court rejected the defendant's argument that the law needed to be easier to apply and more predictable, in essence claiming that the law is about equitable flexibility.[16] Finally, the court rejected the defendant's claims that extending copyright protection would give too much protection to authors and thus stifle innovation and development in the software industry. The court found not only that this was an uncompelling argument,[17] but also that it was largely irrelevant.[18] In this vein, the court specifically rejected the argument that Lotus 1-2-3 was a de facto industry standard and thus not protectable.[19]

II. THE *COMPUTER ASSOCIATES* DECISION

Computer Associates developed a computer program that helped IBM mainframe computer systems schedule various jobs. Part of this program was called "Adapter."[20] In 1982 Altai began marketing a similar

---

13. *Id.* at 54.

14. *See id.* at 60-61.

15. *See id.* at 54-58, 71-72. The defendants based this argument on *Synercom*. *See supra* note 10.

16. *See id.* at 73 (maintaining that a bright-line rule is unnecessary and pointing out that Congress had not adopted such a rule).

17. *See id.* at 73-77 (noting that both sides had presented evidence on the issue and that neither had made a compelling case).

18. *See id.* (concluding that the congressional mandate must be obeyed even if there are policy arguments questioning Congress' wisdom).

19. *See id.* at 78.

20. Computer Assocs. Int'l, Inc. v. Altai, Inc., Nos. 91-7893 & 91-7935, 1992 U.S.

product called "Zeke," and, as the demand for Zeke grew, the president of Altai contacted a programmer at Computer Associates to help update it.[21] This programmer was familiar with Computer Associates's Adapter program and literally copied about thirty percent of Computer Associates's Adapter code into the new version of Altai's program, "Oscar."[22] In 1989 Computer Associates became aware of the copying, secured copyrights to Adapter, and initiated copyright and trade secrets litigation.[23] Reacting to Computer Associates' concerns, Altai used different programmers to rewrite Oscar without utilizing the copied code.[24] Once the rewrite was completed, Altai marketed this second version of Oscar and offered a free upgrade to all users of the old version.[25] The trial court found that Altai had copied Adapter into the original version of Oscar and awarded damages, but held that the second version of Oscar did not infringe Computer Associates' copyrights.[26] Computer Associates appealed the latter ruling to the Second Circuit, which affirmed the trial court's decision.[27]

Writing for the court, Circuit Judge Walker had no difficulty finding that the scheme of copyright protection codified in 17 U.S.C. § 102 provides protection against more than literal copying.[28] The difficult issue for the *Computer Associates* court was determining "the scope of copyright protection that extends to a computer program's non-literal structure."[29] The answer lay in the "venerable doctrines of copyright law," namely the idea-expression dichotomy contained in 17 U.S.C. § 102(b).[30]

The court discussed some of the difficulties of separating copyrightable expression from unprotectable idea,[31] especially when the creative work is of a utilitarian nature, as any computer program will be.[32] While noting that *Baker v. Seldon* is the seminal case dealing with utilitarian works, the court argued that *Baker* "offers scant guidance on how to

---

App. LEXIS 14305, at *10 (2d Cir. June 22, 1992).

21. *See id.* at *13-*15.
22. *See id.* at *14-*15.
23. *Id.* at *15-*16.
24. *Id.* at *16.
25. *Id.* at *16-*17.
26. *See id.* at *4-*5.
27. *See id.* at *5.
28. *See id.* at *23-*24.
29. *Id.* at *25.
30. *Id.* at *26.
31. *See id.* at *26-*36.
32. *See id.* at *28-*29.

separate idea or process from expression, and moreover, on how to further distinguish protectable expression from that expression which 'must necessarily be used as incident to' the work's underlying concept."[33] The "most thoughtful attempt"[34] to actually do so was found in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[35] The court noted that while judicial acceptance of *Whelan* has been mixed, the academic community has widely criticized the decision.[36]

Distinguishing *Whelan*, Judge Walker refused to find one over-arching idea embodied in any computer program.[37] The court's explanation of the programming process,[38] as well as the abstraction test it advanced,[39] evidence its substantial agreement with "[t]he leading commentator in the field," who states that "the crucial flaw in [*Whelan's*] reasoning is that it assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression."[40] The *Computer Associates* court explained that the ultimate function of a computer program is a composite of interacting subroutines, each of which can be viewed as an individual sub-program with its own idea.[41]

To improve upon *Whelan*, the court presented what it called a practical three-step "abstraction-filtration-comparison" approach to determine substantial similarity.

A. Abstraction

As a first step, the structure of the allegedly copied program must be broken down and each level of abstraction isolated so that the court can find the idea(s) at each level.[42] The court thought of this process as tracing the steps which the programmer took to create the program,

---

33. *Id.* at *32 (quoting Baker v. Seldon, 101 U.S. 99, 104 (1879)).
34. *Id.*
35. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).
36. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *33-*34.
37. *See id.* at *34-*35.
38. *See id.* at *5-*10.
39. *See infra* notes 42-44 and accompanying text.
40. *Computer Associates*, 1992 U.S. App. LEXIS 14305, at *34 (quoting 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03[F], at 13-62.34 (1991)).
41. *See id.* at *35.
42. *See id.* at *38-*39 (citing Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930); 3 NIMMER & NIMMER, *supra* note 40, § 13.03[F], at 13-62.34 to 13-63).

except in reverse order.[43] The court offered an "anatomical guide" to the process:

> At the lowest level of abstraction, a computer program may be thought of . . . 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 functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace the implementations of [lower level] modules, . . . until finally, one is left with nothing but the ultimate function of the program . . . .[44]

B. Filtration

Once the levels of abstraction are laid out, a court must examine each level to separate idea from expression.[45] Instead of leaving future courts to start from scratch, the *Computer Associates* court explicitly listed three factors to consider.

First, if the choices made by the programmer were necessary to efficiently implement that part of the program, then the expression has merged with the idea and is not protectable.[46] Judge Walker rationalized this test as a straightforward application of the merger doctrine.[47] Second, the court extended the *scenes a faire* doctrine to computer copyright protection and declared that when a design choice is based on an external factor it is not protected.[48] According to the court, external

---

43. *Id.* at *39.

44. *Id.* at *39-*40 (quoting Steven R. Englund, Note, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Expression of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 897-98 (1990)).

45. *Id.* at *40-*41.

46. *See id.* at *41-*48.

47. *See id.* (citing Baker v. Seldon, 101 U.S. 99 (1879), and its progeny as authority for the merger doctrine and justifying the application of the doctrine here not only because programmers have a special need for efficiency, but also because of the utilitarian nature of their works); *see also* Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp 37, 66 (D. Mass. 1990) (finding that a number of expressive elements in a spreadsheet program merged with the idea of an electronic spreadsheet).

48. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *48-*51 (citing Hoehling v. Universal City Studios, Inc., 618 F.2d 972 (2d Cir.), *cert. denied*, 449 U.S. 841 (1980), as authority for the *scenes a faire* doctrine). As the *Hoehling* court stated in its discussion of certain similarities between literary works: "[t]hese elements, however, are merely *scenes a faire*, that is 'incidents, characters or settings which are as a practical

factors include: (1) mechanical specifications of the computer; (2) compatibility requirements of other programs with which a program is designed to operate; (3) computer manufacturer's design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.[49] It is not clear how these factors are distinguishable from the efficiency concerns listed above, but the court seems to be saying that industry standards are not protectable.[50] Third, material that is already in the public domain is not protectable and must be filtered out.[51]

C. Comparison

After a court has determined which parts of the programmer's decisions are copyrightable, the court must decide if any aspect of the protected expression has been copied.[52] The court added a qualitative element to this test, requiring "an assessment of the copied portion's relative importance with respect to the plaintiff's overall program."[53]

The court rejected the defendant's claim that protection was necessary to supply programmers with enough of an incentive to create software. The court started this analysis by noting that "[t]he immediate effect of our copyright law is to secure a fair return for an 'author's' creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good. . . . [T]he Copyright Act must be

---

matter indispensable, or at least standard, in the treatment of a given topic.'" *Hoehling*, 618 F.2d at 979 (quoting Alexander v. Haley, 460 F. Supp. 40, 45 (S.D.N.Y. 1977)).

49. *Computer Associates*, 1992 U.S. App. LEXIS 14305, at *49 (citing 3 NIMMER & NIMMER, *supra* note 40, § 13.03[F], at 13-66 to 13-71). The *Computer Associates* court also pointed out that some of these factors had already been applied to deny copyright protection to elements of computer programs. *See, e.g.*, Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1262 (5th Cir.), *cert. denied*, 484 U.S. 821 (1987) (finding that externalities of the cotton market dictated organizational similarities between two programs); Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 995 (D. Conn. 1989) (noting that the type of hardware used influenced similarities in the method of screen navigation); Q-Co Indus. v. Hoffman, 625 F. Supp. 608, 616 (S.D.N.Y. 1985) (holding that the program modules in question were an inherent part of any similar program).

50. By doing so, the court appeared to disagree with the *Lotus* court. *See supra* note 19 and accompanying text.

51. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *51-*52; *see also* E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1499 (D. Minn. 1985) (noting that while some aspects of a copied algorithm were in the public domain, the author's contribution of non-trivial original features made the algorithm in question copyrightable).

52. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *52.

53. *Id.*

construed in light of this basic purpose."[54] The court then explained how the Supreme Court's decision in *Feist Publications, Inc. v. Rural Telephone Service Co.*[55] indicated that copyright protection of computer programs should not turn on whether the programmer has expended substantial efforts in creating a product.[56] The court then added that not only was the evidence of "dire consequences" unpersuasive, but also that "serious students of the industry have been highly critical of the sweeping scope of copyright protection engendered by the *Whelan* rule."[57]

The court concluded by noting that the scope of copyright protection was still "not completely clear," explaining that it might only be a "relatively weak barrier."[58] The court briefly argued that copyright law was not well-suited to the task at hand, and suggested that patent law might offer a more acceptable solution.[59]

III. CRITIQUE

A. Repudiation of Whelan *and the Essence of Copyright*

The *Computer Associates* decision clearly rejected the expansive protection of non-literal aspects of computer programs found in *Whelan* and *Lotus*.[60] However, in reaching this decision the court failed to effectively analyze the deeper question: How much protection should the law give to software developers?

The essential function of copyright law is to benefit the public welfare by encouraging authors to create. The constitutional authorization for copyright protection states that the purpose of patent and copyright

---

54. *Id.* at *54 (quoting Twentieth Century Music Corp. v. Aiken, 422 U.S. 151, 156 (1975)).

55. 111 S. Ct. 1282 (1991).

56. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *55-*56 (citing *Feist*, 111 S. Ct. at 1291-95 (holding that originality, not "sweat of the brow," is the key for copyright law)).

57. *Id.* at *56.

58. *Id.* at *57.

59. *See id.* at *57-*59.

60. *Lotus* is widely acknowledged to be a broad extension of copyright protection. *See, e.g.*, Gregory J. Ramos, Note, Lotus v. Paperback: *Confusing the Idea-Expression Distinction and Its Application to Computer Software*, 63 U. COLO. L. REV. 267, 267 (1992). The *Computer Associates* court recognized that its holding might cut back on the scope of copyright protection for non-literal aspects of computer programs: "If the test we have outlined results in narrowing the scope of protection, *as we expect it will* . . . ." *Computer Associates*, 1992 U.S. App. LEXIS 14305, at *58-*59 (emphasis added).

protection is "[t]o promote the Progress of Science and useful Arts."[61] Congress followed this mandate when it drafted the Copyright Act. The addition of computer programs to the Copyright Act in 1980 served to bring software within the policy of using private benefits to advance the public welfare. The judiciary has agreed with this approach.[62] Academia has also stressed the importance of correctly determining the appropriate extent of private benefit. A major criticism of *Lotus* is that broad protection ignores the economic realities of the programming world.[63] This debate is not merely academic; segments of the programming community protested the *Lotus* decision because it had dangerous chilling effects on creativity.[64]

Unfortunately, determining the amount of protection necessary to set the correct incentives for authors to create, and thereby benefit the public, is a difficult problem that has been largely ignored. Dependable answers are elusive. For example, courts have commented that the ultimate effects of their decisions concerning protection are unclear and open to debate.[65] In addition, much of the scholarly analysis has been limited to establishing a framework to analyze the question.[66] Consequently, there is a remarkable dearth of empirical evidence as to the actual effects of protection.

While recognizing the underlying incentives argument, the *Computer Associates* court refused to address directly Computer Associates' earnest

---

61. U.S. CONST. art. I, § 8, cl. 8.

62. *See, e.g.*, Mazer v. Stein, 347 U.S. 201, 219 (1954) ("The economic philosophy behind the clause . . . is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare . . . .").

63. *See, e.g.*, Ramos, *supra* note 60, at 267 (stressing the undesirable effects *Lotus* will have on standardization and innovation); Karen S. Kovach, Note, *Computer Software Design: User Interface - Idea or Expression?*, 60 U. CIN. L. REV. 161, 185-86, 190 (1991) (expressing concern that software companies will use the courts to stop competitors).

64. *See, e.g.*, Gus Venditto, *Pipeline Column*, PC COMPUTING, Oct. 13, 1992, at 30 ("[i]t's clear that a chilling effect has already taken place" as a result of Lotus's litigation efforts).

65. *See, e.g.*, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 74-76 (D. Mass. 1990) (explaining that both the plaintiff and defendant had presented "sharply contrasting" evidence on the general effect protection would have on incentives); Computer Assocs. Int'l, Inc. v. Altai, Inc., Nos. 90-7893 & 91-7935, 1992 U.S. App. LEXIS 14305, at *56 (2d Cir. June 22, 1992) (indicating that the evidence on incentives was inconclusive).

66. *See, e.g.*, Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1058-71 (1989) (offering an "economic framework for analyzing legal protection for application programs"); John S. Wiley, Jr., *Copyright at the School of Patent*, 58 U. CHI. L. REV. 119, 138-44 (1991) (arguing that both patent and copyright law should utilize a consumer-oriented approach to reach their common goal of rewarding innovation to encourage the creation of new information).

efforts to resolve this debate on its merits.[67] Instead, Judge Walker followed the lead of the *Lotus* court and maintained that the level of protection was set by the legislature.[68] To buttress this position, the *Computer Associates* court claimed that because *Feist* repudiated the "sweat of the brow" argument for protection, courts are no longer concerned about protecting authors' labors—they just apply the law.[69]

These justifications for avoiding the incentives debate are unpersuasive. While it is not clear that courts have the ability or the desire to set correctly the balance between protecting software authors and preventing monopolies, they should not ignore the issue or down-play its importance. As noted above, incentives form the essence of the copyright puzzle. Simply put, copyright protection exists to advance the public welfare through the creation and availability of software. There is little solid ground on which to base a well-reasoned copyright decision if this essential tension is ignored. Even though the correct level of incentives is difficult to determine, courts should openly analyze the question. Ignoring the effects that a decision will have on the flow of computer software only advances ignorance. To say that the appropriate level of protection is unclear is not an answer to the question before the court; it is only the starting point of the necessary inquiry.[69]

Despite the apparent difficulties involved in addressing the incentives debate, courts can effectively analyze the issue. In *Sega Enterprises LTD. v. Accolade, Inc.*,[70] the Ninth Circuit grappled with the incentives puzzle.[71] In determining whether disassembly of Sega's program was a fair use, for example, the court relied in part on the underlying incentives involved. While noting that protecting expression encourages production of original works, the court argued that in this case disassem-

---

67. *See Computer Associates*, 1992 U.S. App. LEXIS 14305, at *53-*59.

68. *See id.* at *55-*57.

69. In keeping with its tone of legislative deference, another rationale the *Computer Associates* court gave for avoiding the incentives debate was that if more protection for software developers is necessary, then the legislature could consider shifting to patent-like protection. *See id.* at *57-*59. The court refrained from discussing a patent approach in any depth because of a clear congressional mandate to protect software under copyright. *See id.* at *58. The possibility of legislative overhaul of the copyright system does not justify the court's avoidance of the entire incentives issue. If the court felt that various aspects of copyright could benefit from a patent-like bent, then it could have attempted to integrate the two. *See, e.g.*, Wiley, *supra* note 66, at 144-80 (critically analyzing the differences between patent and copyright protection).

70. No. 92-15655, 1992 WL 293141 (9th Cir. Oct. 20, 1992).

71. *See id.* at *10-*12, *15-*16.

bly was the only way to access the ideas in the program.[72] The public benefit gained by having those ideas in the public domain for others to build on, the court contended, supported a finding that disassembly was fair use.[73] Thus, by directly discussing the purpose, effect, and character of the copying, the court dealt with the underlying purpose of copyright protection and demonstrated that courts can effectively analyze this issue.

While the *Computer Associates* court might well have made the right decision when it reigned in the runaway copyright doctrine, the relative lack of reasoned analysis about the effect this decision would have on incentives for the design and production of computer software was a serious omission. For decisions like *Computer Associates* to make sense in the real world, courts will have to address and answer, to the best of their abilities, these questions which are at the base of the copyright riddle.

B. Examination of Abstraction-Filtration-Comparison

Not only did *Computer Associates* explicitly reject *Whelan*,[74] but it also undercut the *Whelan* rationale by de-emphasizing the role of the idea-expression distinction in the determination of copyrightability. Instead of being the basis for the decision, the idea-expression dichotomy was used by the *Computer Associates* court as a way to organize the ultimate inquiry. This transformation occurred in the first step of the abstraction-filtration-comparison test, where the realization that the idea-expression distinction necessarily turned on the definition of a level of abstraction served as the justification for decomposing the program in the same way as the programmer broke down the original problem.[75]

Decomposing a program in this manner is a major improvement over the *Whelan* approach. It has been clear since *Nichols* that the different levels of abstraction inherent in any idea make separating idea from expression extremely difficult.[76] Remarkably, the process that programmers go through in creating their program supplies a conceptually clear

---

72. *See id.* at *15-*16.
73. *See id.*
74. *See supra* notes 37-41 and accompanying text.
75. *See supra* notes 42-44 and accompanying text.
76. *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930); *see also* Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) (arguing that such decisions are inevitably ad hoc).

answer: A court should examine each of the levels that a programmer would. Since the programmer moves through a variety of levels of expression, the somewhat arbitrary fixation on one particular level becomes unnecessary.

The parallels between program design and Judge Hand's quest are too striking to ignore. While recreating a programmer's path is undoubtedly difficult, the court's reference to reverse engineering[77] suggests that it could be done by someone well-versed in programming. This concept of abstraction is familiar to all computer science students.[78] One common method of program design is to start at a very general level—defining the task the program is supposed to undertake—and then breaking this general task down into sub-problems. The process of breaking sub-problems into further sub-problems continues until the programmer has discrete, manageable tasks to code. The ultimate number of layers will depend on the complexity of the task.

Despite the apparent link between the abstraction step and programming methodology, judicial acceptance of the *Computer Associates* test has been mixed. In *Atari Games Corp. v. Nintendo, Inc.*,[79] the court applied the abstraction method to separate a program into manageable components.[80] In contrast, in *Lotus Development Corp. v. Borland International, Inc.*,[81] the same court that decided the earlier *Lotus* case essentially looked for the one overarching idea of the program. While the court noted that a program could have more than one idea,[82] it seemed to view its task as choosing among competing formulations of *the* idea behind the program.[83] Although the *Borland* court purported to essentially agree with *Computer Associates*,[84] the difference in methodologies is significant. By not using the idea-expression distinction as an organizational scheme, the court perpetuates the myth that invoking the idea-expression distinction somehow provides insight into separating protectable aspects of programs from unprotectable ones. In the end, the *Borland*

---

77. *See Computer Assocs. Int'l, Inc. v. Altai, Inc.*, Nos. 90-7893 & 91-7935, 1992 U.S. App. LEXIS 14305, at *39 (2d Cir. June 22, 1992).

78. *See, e.g.*, CLEMENT L. MCGOWAN & JOHN P. KELLY, TOP-DOWN STRUCTURED PROGRAMMING TECHNIQUES, 6-15, 93-100 (1975); NICKLAUS WIRTH, ALGORITHMS + DATA STRUCTURES = PROGRAMS, xii-xiii, 1-4 (1976) (introducing abstract data types).

79. No. 91-1293, 1992 WL 217828 (Fed. Cir. Sept. 10, 1992).

80. *Id.* at *5.

81. No. 90-11662-K, 1992 U.S. Dist. LEXIS 11358 (D. Mass. July 31, 1992).

82. *See id.* at *35-*36.

83. *See id.* at *21-*22, *39-*42.

84. *See id.* at *23-*25.

court had to go beyond idea-expression and relied heavily on what the *Computer Associates* court termed filtration to determine what was protectable.[85]

Much of the most important analysis in the *Computer Associates* methodology occurs when each of these layers of abstraction is examined and the unprotectable ideas filtered out.[86] Consequently, the real question is whether its tripartite inquiry into efficiency, external factors, and the public domain is an improvement over the previous case law. Filtration, as envisioned by the *Computer Associates* court, was intended to result in less protection for software. The inquiry into efficiency and external constraints emphasizes the practical constraints placed on any programmer by the utilitarian nature of the task.[87] Since programming is largely a functional process, such an inquiry will likely result in narrower protection.[88]

While focusing attention on an important aspect of the problem, the inquiry into efficiency raises some troubling questions because efficiency is not always easily or accurately measured.[89] In addition, because efficiency is often measured on a very coarse scale, this test might not offer much guidance in close cases, thus forcing a judge to decide if an aspect of a program is "efficient enough" to prevent protection.[90] Furthermore, such a standard will encourage plaintiffs to overwhelm the judge with "other" efficient approaches that an alleged infringer could have utilized, while the defendants will explain why these alternatives were not viable. Unless the judge is well-versed in software engineering, the possibilities for confusion abound.

---

85. The *Borland* court ends up doing most of its work when it tries to determine if the expression of "the" idea is limited to essential elements. *See id.* at *22, *43-*49.

86. *See Computer Assocs. Int'l, Inc. v. Altai, Inc.*, Nos. 91-7893 & 91-7935, 1992 U.S. App. LEXIS 14305, at *41 (2d Cir. June 22, 1992) ("Strictly speaking, this filtration serves 'the purpose of defining the scope of plaintiff's copyright.'") (quoting Brown Bag Software v. Symantec Corp., No. 89-16239, slip op. 3719, 3738 (9th Cir. Apr. 7, 1992)).

87. *See id.* at *43-*45.

88. *See* Sega Enterprises LTD. v. Accolade, Inc., No. 92-15655, 1992 WL 293141, at *13 (9th. Cir. Oct. 20, 1992) ("Under a test that breaks down a computer program into its component subroutines and sub-subroutines and then identifies the idea or core functional element of each, such as the [*Computer Associates* test], many aspects of the program are not protected by copyright.").

89. *See, e.g.*, SARA BAASE, COMPUTER ALGORITHMS - INTRODUCTION TO DESIGN AND ANALYSIS 15-46 (2d ed. 1988); 1 DONALD E. KNUTH, THE ART OF COMPUTER PROGRAMMING - FUNDAMENTAL ALGORITHMS 94-102 (2d ed. 1973).

90. Computer scientists measure the efficiency of a given solution (or algorithm) by doing complex asymptotic calculations to determine, on an order of magnitude, roughly how long it will take. *See* BAASE, *supra* note 89, at 28-35; KNUTH, *supra* note 89, at 104-19.

The second factor in filtration, external constraints, is even more amorphous than efficiency. Despite the list of five sample factors given,[91] any lower court applying this filter will be left largely to its own means. One way to interpret this filter is as a catch-all, covering other reasons besides efficiency that a program might be similar to another. The sample "externalities" listed suggest one litmus test, that of commercial viability.[92] At one extreme, the test could mean that any commercially successful product must have been tailored to fill a specific niche, and thus was dictated by external factors (and therefore unprotected). It is difficult to determine where to draw this business necessity line in a principled manner. Moreover, this raises the concern that the protectability of a software package will turn on factors like the structure of the market, in effect requiring an analysis of the necessary protection required to encourage the production of software.[93] One worry of the *Lotus* court seems particularly appropriate: Does the copyrightability of a program change when it becomes commercially successful?[94] If the existence of a large and successful competitor is enough to create a standard, then the fears of the *Lotus* court might well be justified.[95]

Recent judicial opinions reinforce the conclusion that the *Computer Associates* test will be difficult to apply. In *Borland*, for example, the judge seemed to reduce the complex issue of whether external constraints necessitated similarities between two programs into a somewhat absurd combinatorial argument.[96] While the existence of other commercially successful interfaces might indicate that imitation was not sufficiently

---

91. *See supra* notes 48-49 and accompanying text.

92. In order to be conceptually distinct from efficiency, these factors must test some other reason why users prefer a product. A loose way to define this preference is commercial viability.

93. It is noteworthy that the court used *Feist* to explicitly reject analysis of this sort. *See supra* notes 55-57 and accompanying text.

94. *See* Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 78-79 (D. Mass. 1990) (explicitly rejecting the standardization argument).

95. *But see* Walter G. Duflock, Note, *"Look and Feel": A Proposed Solution to the Diverging Views Between the Software Industry and the Courts*, 8 SANTA CLARA COMPUTER & HIGH TECH. L.J. 447, 458-64 (1992) (contending that protection of non-literal elements of programs adversely affects standardization and innovation); Kovach, *supra* note 63, at 185-86 (arguing that protection of functional aspects of software prevents standardization and harms the public).

96. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., No. 90-11662-K, 1992 U.S. Dist. LEXIS 11358, at *43-*47 (D. Mass. July 31, 1992) (arguing that because other choices and arrangements of words in a menu system were possible, the given arrangement must not have been necessary). This argument ignores much of the subtle and complex controversy that *Computer Associates* termed external constraints.

important,[97] the recognition of this point does not seem to trump further inquiry into what similarities were necessitated by the market. However, because there is no easy answer to the deeper questions regarding incentives, the court might have felt that snappy rhetorical arguments were adequate. This unfortunate state of affairs is traceable to the *Computer Associates* court's inability to explain how to conduct its suggested inquiry. *Atari v. Nintendo*[98] also provides some insight into the difficulty of applying the Computer Associates filtration test. Because the defendant had misappropriated the plaintiff's program in a rather egregious manner, a detailed inquiry into the intricacies of copyright law was probably unnecessary.[99] In any event, the court quickly determined that no external factors were present and that the elements copied were not necessary for the implementation of the larger program.[100] The summary nature of this determination indicates the "seat-of-the-pants" nature of the *Computer Associates* test. While a capable jurist might be able to know infringement when she sees it, the important questions raised by *Computer Associates* do not get carefully examined.

Sega v. Accolade, on the other hand, provides some evidence of a judicial ability to analyze these difficult questions effectively. That court focused on the fair use defense to a copyright infringement action in order to examine factors somewhat similar to the *Computer Associates* filtration test.[101] While using a wider frame of reference than found in *Computer Associates*, the *Accolade* court's deft handling of these sticky economic issues provides some hope that judges can manage the sort of analysis that *Computer Associates* calls for.

In short, while proffering a pragmatic approach to the problem, the *Computer Associates* court has done little more than create a procedural framework upon which it draped a vague overview of current copyright law. Future courts applying these filters will have little other than the current case law regarding the idea-expression dichotomy on which to fall back. With time, the court's abstraction-filtration-comparison test may provide the framework for a careful inquiry into whether a copyrighted

---

97. *See id.* at *43 (noting that Borland's "Quattro Pro" spreadsheet uses a different menu tree).

98. Atari Games Corp. v. Nintendo, Inc., No. 91-1293, 1992 WL 217828 (Fed. Cir. Sept. 10, 1992).

99. *See id.* at *1-*2, *8, *10-*11.

100. *See id.* at *5-*6.

101. *See* Sega Enterprises LTD. v. Accolade, Inc., No. 92-15655, 1992 WL 293141, at *10-*13 (9th. Cir. Oct. 20, 1992).

work has been infringed, but for now the opinion offers little substantive guidance for a lower court to follow beyond its categorical imperative to reduce protection. In addition, the court's desire to avoid balancing an initial software designer's need for protection against the competing needs of those who follow him or her indicate that an important factor in the underlying decision will be omitted.