Harvard Journal of Law & Technology
Volume 9, Number 2 Summer 1996

COMPUTERS, COPYRIGHT, AND FUNCTIONALITY: THE FIRST CIRCUIT'S DECISION IN *LOTUS DEVELOPMENT CORP. V. BORLAND INTERNATIONAL, INC.*

Robert L. Bocchino Jr.[*]

I. INTRODUCTION

In *Lotus Development Corp. v. Borland International, Inc.*,[1] the U.S. Court of Appeals for the First Circuit ruled that the menu structure of Lotus 1-2-3 is a "method of operation" and thus unprotected by copyright.[2] The First Circuit's decision was affirmed by a divided Supreme Court, so this result is now firmly a part of copyright law.[3] The Supreme Court, however, issued no written opinion in the case; thus any clues regarding the ramifications of this important case in software copyright must be found, if at all, in the First Circuit's decision. Unfortunately, while the First Circuit ably treated the policy considerations underlying the case, it failed to provide a coherent theoretical justification for its result. Thus, *Borland* will likely be of little help in resolving the difficult questions that will inevitably arise regarding the proper scope of software copyright, an area which has continued to vex the courts.

This Note argues that copyright protection should extend only to the implementation of program functionality — the program code and its attendant non-literal elements such as program structure — and that the functionality itself should be unprotected. This understanding of software copyright protection represents the only viable way to reconcile the traditional underpinnings of copyright law with the intent of Congress as embodied in the report by the National Commission on New Technological Uses ("CONTU").[4] In particular, the First Circuit's result

---

* J.D., Harvard Law School, Class of 1997.

1. 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 116 S. Ct. 804 (1996).

2. Lotus 1-2-3 is a spreadsheet program which employs a graphical user interface containing what programming parlance refers to as pull-down menus. The user interacts with the program by selecting items from the menus, which may uncover other menus with further items. The user navigates through the menu structure until the desired functionality is activated. *See Borland*, 49 F.3d at 809.

3. Lotus Dev. Corp. v. Borland Int'l, Inc., 116 S. Ct. 804 (1996), *aff'g by an equally divided Court*, 49 F.3d 807 (1st Cir. 1995).

4. NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1978). In 1974, Congress formed CONTU to study, among other things, the problem of software copyright. *See* 1 MELVILLE B. NIMMER & DAVID NIMMER,

is correct: user interfaces of the type at issue in *Borland* should not be the subject of copyright protection. This result is further buttressed by the fundamental inconsistency of protecting the user interface of a general-purpose computer running a program while leaving unprotected the interfaces of other machines, such as dedicated-purpose computers and mechanical devices.

II. THE DISTRICT COURT DECISION

Borland first marketed its Quattro spreadsheet program in 1987.[5] Quattro's creators designed the program to include significant innovations over then-existing spreadsheet programs, including Lotus 1-2-3, which was the dominant spreadsheet on the market at that time. To attract existing 1-2-3 users, who had presumably expended significant effort mastering 1-2-3's user interface, Borland made its program compatible with 1-2-3 by including a copy of 1-2-3's menu structure in its Quattro and Quattro Pro programs. It included this copy in two forms: the Lotus Emulation Interface and the Key Reader. By activating the Lotus Emulation Interface, the Quattro user could interact with the program through a version of the Lotus menu structure.[6] The Key Reader was a translation program enabling Quattro and Quattro Pro to execute any macros the user had written for 1-2-3.[7] Thus, armed with the Lotus Emulation Interface and Key Reader, software purchasers could switch from Lotus's to Borland's program without learning a new interface or writing new macro programs. The user could also choose to ignore the Lotus Emulation Interface and Key Reader, and use Quattro

---

NIMMER ON COPYRIGHT § 2.04[C], at 2-51 to 2-52 n.21 (1995). There is some disagreement regarding the extent to which the CONTU report should inform the interpretation of the provisions in the 1976 Copyright Act concerning computer software. Because Congress adopted CONTU's recommendations for software protection virtually unaltered, however, at least some courts have indicated that the CONTU report should be treated as legislative history. *See* 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, 886 (1990). This approach seems reasonable, given the paucity of clues to the nature of software protection provided in the Act itself and in the House Report.

5. *See Borland*, 49 F.3d at 810.

6. *See id.* In Lotus Emulation Mode, the screen appeared slightly different from 1-2-3, and certain Borland options were disabled. *See id.*

7. A macro is a recorded sequence of menu commands. Instead of retyping the commands every time they are needed, the user can activate the sequence by striking a single key. *See id.* at 809. For frequently used sequences, this can represent a significant increase in efficiency. *See id.* at 809-10. The macro itself is thus like a computer program, written in the "language" of the menu interface.

in its "native mode."[8] Borland did not copy any of Lotus's computer code in incorporating the 1-2-3 interface into its program.[9]

Subsequent to Quattro's release, in *Lotus Development Corp. v. Paperback Software International, Inc.*,[10] Judge Keeton of the District of Massachusetts ruled that the menu structure was expression protected by Lotus's copyright in its 1-2-3 program. Judge Keeton based this result on the apparently large number of similar interface arrangements, and on the creativity which evidently went into Lotus's design.[11] In light of this ruling, Borland filed an action for declaratory judgment of non-infringement in the District Court for the Northern District of California. Lotus then filed the instant action for copyright infringement in the District of Massachusetts, and the Northern District of California dismissed Borland's declaratory judgment action in favor of the Massachusetts action. Judge Keeton presided over this litigation as well.[12]

Both parties filed cross motions for summary judgment, which the district court denied.[13] Judge Keeton, however, allowed the parties to file renewed motions for summary judgment in light of his rulings on the first motions. Borland's second motion contended that the Lotus 1-2-3 menus were unprotected by copyright, and that no reasonable trier of fact could find actionable similarity between the programs. Lotus, on the other hand, contended that Borland had copied its entire interface and had thus infringed its copyright.[14]

Ruling on this second pair of motions, Judge Keeton denied Borland's motion and granted Lotus's motion in part.[15] Judge Keeton concluded that the test for copyrightability he had announced in his earlier *Borland* ruling[16] was consistent with the "abstraction-filtration-comparison" analysis subsequently embraced by the Second Circuit in *Computer Associates International, Inc. v. Altai, Inc.*[17] Applying his test

---

8. *See id.* at 810; *see also* Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 206 (D. Mass. 1992).

9. *Borland*, 49 F.3d at 810.

10. 740 F. Supp. 37 (D. Mass. 1990).

11. *See Paperback*, 740 F. Supp. at 56, 67.

12. *Borland*, 49 F.3d at 810 n.1.

13. Lotus Dev. Corp. v. Borland Int'l, Inc., 788 F. Supp. 78, 80 (D. Mass. 1992).

14. *See Borland*, 49 F.3d at 810.

15. Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 205 (D. Mass. 1992).

16. *See Borland*, 788 F. Supp. at 89-90.

17. 982 F.2d 693 (2d Cir. 1992). In *Altai*, the Second Circuit applied the famous test for literary works articulated by Judge Learned Hand in Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930), and considered the program at various levels of abstraction to ascertain where protected expression crossed into the realm of unprotected idea. *See generally* 3 NIMMER & NIMMER, *supra* note 4, § 13.03[F]. The court agreed with the District Court's division of the program into the following levels, from least to most general:

to the elements of the 1-2-3 interface, Judge Keeton concluded that "very satisfactory" menus could be constructed by varying either the menu structure or the command words on the menus,[18] and that Borland had thus infringed as a matter of law.[19] Nevertheless, he set a jury trial to determine the scope of infringement, including the extent of the copying and the protectability of Lotus's "long prompts,"[20] and to consider the possibility of functional constraints on the menu arrangement.[21] Following this ruling, Borland duly removed the Lotus Emulation Interface from its Quattro products. It retained the Key Reader, however, so that the Quattro programs could still run Lotus macros. Judge Keeton permitted Lotus to file a supplemental complaint alleging infringement based continued use of the Key Reader.[22]

The parties agreed to a bench trial on the remaining issues.[23] Judge Keeton held two trials, one for the Emulation Interface and one for the Key Reader. At the close of the Phase I trial, Judge Keeton allowed Borland to amend its answer to include the affirmative defense of fair use.[24] Judge Keeton rejected this defense. In his Phase II decision, Judge Keeton found that the Key Reader constituted a "virtually identical copy of the Lotus menu tree structure" and thus infringed Lotus's protectable expression.[25] Judge Keeton permanently enjoined Borland from including the Key Reader in its products.[26]

---

object code, source code, parameter lists (the formats in which information is passed between modules of the program), services required (program functionality), and general program outline. *Altai*, 982 F.2d at 714.

An important distinction not mentioned by Judge Keeton, however, is that both *Altai* and *Nichols* applied the abstractions test with the core of protected material at the lowest level of abstraction. In *Borland*, by contrast, though the question was whether the interface was protected, Judge Keeton's analysis put the interface itself at the lowest level. The First Circuit refused to apply the *Altai* test, essentially on this ground. *See infra* notes 31-34 and accompanying text.

18. *See Borland*, 799 F. Supp. at 217.

19. *Id.* at 223.

20. Lotus 1-2-3's long prompts consist of text displayed on the screen when a user passes over a menu command, explaining what that command will do if selected. *See Borland*, 49 F.3d at 811 n.2.

21. *See Borland*, 49 F.3d at 811. Judge Keeton also resolved additional issues not pertaining to the copyright question.

22. *Id.* at 812.

23. *See id.*

24. *See id.* Fair use is an equitable doctrine, codified at 17 U.S.C. § 107 (1995), under which an otherwise infringing copy may not be actionable. *See generally* 3 NIMMER & NIMMER, *supra* note 4, § 13.05.

25. Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 223, 228 (D. Mass. 1993).

26. *Borland*, 49 F.3d at 812.

III. THE FIRST CIRCUIT'S DECISION

The First Circuit reversed. Writing for a unanimous three-judge panel, Judge Stahl first noted that the sole question on appeal was "whether the Lotus menu command hierarchy is copyrightable subject matter," and that this was a matter of first impression in the First Circuit.[27] Judge Stahl rejected Borland's contention that the result here followed immediately from the Supreme Court's decision in *Baker v. Selden*,[28] since both cases involved accounting systems.[29] Concluding that the question concerned "literal copying of the Lotus menu command hierarchy,"[30] Judge Stahl then declined to apply the abstraction-filtration-comparison test embraced in *Computer Associates International, Inc. v. Altai*.[31] According to the court, *Altai* is of "little help" in passing on the copyrightability of a menu hierarchy.[32] This is so, the court noted, because the abstraction process itself seems to presuppose some lowest level of protectable expression.[33] Thus, applying the abstractions test here would have begged the question and "obscure[d] the more fundamental question of whether a menu command hierarchy can be copyrighted at all."[34]

Instead, the court rested its analysis on the idea that a "'method of operation,' as that term is used in § 102(b), refers to the means by which a person operates something."[35] That is, because the 1-2-3 interface "serves as the method by which the program is operated and controlled,"[36] it is uncopyrightable under § 102(b). The court distinguished methods of operation under this view from such expression as inheres in long prompts and graphical displays, which are incidental to

---

27. *Id.* at 813.

28. 101 U.S. 99 (1879). In *Baker*, the Court held that the copyright in a book describing an accounting system did not extend to protection of the system itself.

29. *See Borland*, 49 F.3d at 814 ("Lotus does not claim to have a monopoly over its accounting system. Rather, this appeal involves Lotus's monopoly over the commands it uses to operate the computer.").

30. *Id.*

31. Computer Associates International, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992). For a discussion of the *Altai* test and its relationship to the district court's decision and accompanying text, see *supra* note 17.

32. *Borland*, 49 F.3d at 815.

33. In fact, the test is usually applied to the core of protected material at the lowest level of abstraction. *See supra* note 17.

34. *Borland*, 49 F.3d at 815.

35. *Id.* 17 U.S.C. § 102(b) (1994), among other things, codifies the holding in *Baker v. Selden* by providing, "In no case does copyright protection . . . extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery."

36. *Borland*, 49 F.3d at 815.

the method and may be separately copyrightable.[37] Because an interface may be encoded in various ways, the court also distinguished the interface itself — the unprotectable method of operation — from the "underlying computer code" implementing it. The First Circuit criticized the district court for "limit[ing] Lotus 1-2-3's 'method of operation' to an abstraction."[38] Thus, as in declining to apply *Altai* approach, the court made it clear its view that the idea/expression dichotomy of copyright law and its levels of abstraction analysis is inapplicable to the question of user interfaces.

The court then justified its reading of § 102(b) in terms of case precedent and public policy. First, since "Lotus wrote its menu hierarchy so that people could learn it and use it,"[39] the result here follows from the general rule in *Baker*, codified at § 102(b): the copyright in a description of a useful art does not extend to a monopoly in the art itself, which can be protected only by patent.[40] Next, the court offered an analogy to the arrangement of buttons on a VCR. According to the court, VCR buttons are the "'method of operating' the VCR," and 1-2-3's menu command terms are "equivalent to the [VCR] buttons themselves," since the 1-2-3 user interacts with the program by selecting menu command terms in the same way that a VCR user accesses the desired function by pushing the VCR buttons.[41] Although VCRs, unlike computer programs, are useful articles and thus outside the ambit of copyright protection, the court explained that because the VCR's button arrangement is a method of operation, it would be unprotected even if the VCR were itself copyrightable.[42] The court also invoked the principle of program compatibility, or "interoperability," as copyright protection for the user interface here would have the undesirable effect of preventing competing software manufacturers from offering compatible products, thus forcing software users to learn multiple interfaces and to write multiple sets of macros:[43] "[T]he user would have to learn not just one method of operating the computer . . . but many different methods. We find this absurd."[44] The majority concluded by asserting the consistency of its approach with that

---

37. *See id.* at 816. For a discussion of the Lotus 1-2-3 long prompts, see *supra* note 20. The court noted that Lotus claimed no copyright infringement of the 'look and feel' of 1-2-3's graphical user interface. *See Borland*, 49 F.3d at 816 n.10.

38. *Borland*, 49 F.3d at 816.

39. *Id.* at 817.

40. *See id.*; Baker v. Selden, 101 U.S. 99, 105 (1879) ("The object of the one is explanation; the object of the other is use.").

41. *See Borland*, 49 F.3d at 817.

42. *See id.*

43. *See id.* at 817-18.

44. *Id.* at 818.

of *Feist Publications, Inc. v. Rural Telephone Service Co.*[45] The court noted that the "right [of authors] to their original expression" is subject to the exclusions in § 102(b).[46] Furthermore, by contrast with most categories of works, the underlying ideas of which can be conveyed effectively without copying protected expression, the policy noted in *Feist* of allowing authors to "build freely upon the ideas and information conveyed by a work"[47] in this context requires the use of "the precise method of operation already employed."[48]

Judge Boudin wrote a concurring opinion further elaborating on the interoperability aspect of Judge Stahl's analysis. The problem with copyright protection for utilitarian articles like computer programs, Boudin noted, is that there is strong public policy interest, embedded in the patent law, in providing protection for such articles for a much shorter term and only on a showing of novelty and nonobviousness. In the case of user interfaces, there is an additional problem: the notorious "switching effect" that occurs when a user's investment in learning a particular interface dissuades the user from replacing it with another one, even if the second interface is superior.[49] Due to the functional aspects of software, applying copyright to computer programs is like "assembling a jigsaw puzzle whose pieces do not quite fit."[50] In light of these problems, Judge Boudin cautioned against the "cookie cutter fashion"[51] in which some courts have adapted traditional copyright notions to computer programs "as if [they] were novels or play scripts."[52] Instead, Judge Boudin argued for a continuation of the case-by-case approach that he says has always characterized the development of copyright law.[53]

Applying these ideas to the facts of the instant case, Judge Boudin agreed with the majority's result. He noted first that Quattro was not simply a clone copy of 1-2-3. Rather, its success was due to the original features it contained; thus Borland's asserted reason for copying the

---

45. 499 U.S. 340 (1991).

46. *Borland*, 49 F.3d at 818 (quoting *Feist*, 499 U.S. at 349-50).

47. *Feist*, 499 U.S. at 350.

48. *Borland*, 49 F.3d at 818. The court also noted that its approach was inconsistent with that in Autoskill, Inc. v. National Educ. Support Sys., Inc., 994 F.2d 1476 (10th Cir.), *cert. denied*, 114 S. Ct. 307 (1993).

49. *See Borland*, 49 F.3d at 819-20 (Boudin, J., concurring) (citing the paradigmatic example in this connection, the QWERTY keyboard, which has been mastered by so many legions of typists that replacing it, even for a superior arrangement, would be all but impossible). *But see* S.J. Liebowitz & Stephen E. Margolis, *Should Technology Choice Be a Concern of Antitrust Policy*, 9 HARV. J.L. & TECH. 283 (1996) (dismissing the QWERTY example as myth).

50. *Borland*, 49 F.3d at 820.

51. *Id.*

52. *Id.*

53. *See id.*

interface seemed genuine. Additionally, there is an important policy interest in allowing just this type of copying: though the 1-2-3 interface may indeed have been creative in some aspects, in others the choices were undoubtedly arbitrary — one can envision many similar interfaces that would have worked just as well.[54] Thus the economic value of the copyright for which Lotus was vying lay to a significant extent not in the exclusive right to employ a new and useful method that patent law traditionally grants, but simply in the inertia inherent in the investment Lotus users had made in learning this particular interface. According to Judge Boudin, if Borland has produced a better product, it should be able to compete with Lotus in a market untrammeled by this switching or network effect.[55]

Thus, Judge Boudin says with unabashed consequentialism, the only question at issue is the choice of a doctrinal hook to justify the majority's result. Judge Boudin noted that the majority's formulation is defensible on the basis of the dictionary definitions of the words "method" and "operate."[56] He also suggested the alternative approach of fair use. The attraction of this approach for Judge Boudin is that it preserves the case-specific analysis required by his concerns; since fair use is an equitable doctrine, if Borland had simply cloned Lotus's program, infringement could have been found on that basis. Judge Boudin noted, however, that applying the fair use doctrine in this connection may be problematic: in addition to the Supreme Court's pronouncement that commercial use is "presumptively . . . unfair,"[57] basing interface protection on fair use would involve the administrative costs and unpredictability inevitably attending this notoriously fact-intensive analysis. Thus, Judge Boudin finds the majority's formulation "as good, if not better, than any other . . . within the reach of courts."[58]

---

54. *See id.* at 821.

55. *See id.*

56. *See id.*

57. Harper & Row Publishers, Inc. v. Nation Enters., 471 U.S. 539, 562 (1985) (quoting Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 451 (1984)). *But see* Campbell v. Acuff-Rose Music, Inc., 114 S. Ct. 1164, 1171 (1994) (indicating that the commercial/non-commercial distinction is to be applied with a view to the purposes of copyright); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992) (holding the fair use doctrine applicable in the software reverse engineering context).

58. *Borland*, 49 F.3d at 822. Judge Boudin also hinted that it might be appropriate for Congress to enact solutions beyond the authority of the courts, such as a shorter period for software copyright. *See id.*

IV. DISCUSSION

The crucial flaw in the First Circuit's analysis is the court's failure to base its result upon a coherent theory founded in traditional copyright principles. This shortcoming is understandable, given the tremendous confusion in the circuits regarding the scope of software copyright protection.[59] Yet analysis of the question presented in *Borland*, as with any question about software copyright, must begin with some apprehension of the vexing and largely unresolved problem which has plagued the courts since the adoption of the 1976 Act: how to make sense of Congress's seemingly paradoxical declaration that computer programs are to be treated as literary works.[60] As argued below, the First Circuit was correct in rejecting the *Altai* test as the starting point of its analysis.[61] The court, however, neither adopts nor articulates anything to be relied on in its stead. Rather, the majority simply begins with the assertion that a "'method of operation' . . . refers to the means by which a person operates something,"[62] and proceeds to support this assertion with various policy rationales which, though generally persuasive, embrace no coherent approach to the underlying problem. Though Judge Boudin's observation about the ill-fitting puzzle pieces of software copyright is well taken, an ad hoc approach devoid of any overarching theoretical framework inevitably produces inconsistency and unpredictability. To the extent possible, then, courts should attempt to provide a blueprint for resolving the hard questions which will inevitably arise in the continuing struggle to apply copyright principles to program functionality.

In addressing this problem in the future, courts should base their analysis on a theoretical framework that limits copyright to protecting the way in which program functionality is encoded, while leaving the functionality itself in the public domain to be freely implemented in other programs, unless protected by patent. This approach was presented to the First Circuit by several of the amici curiae[63] and finds support in

---

59. *See, e.g.*, Irwin Gross, *A New Framework for Software Protection*, 20 RUTGERS COMPUTER & TECH. L.J. 107, 132-42 (1994).

60. *See* H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5666.

61. *See infra* note 75.

62. *Borland*, 49 F.3d at 815.

63. *See* Dennis S. Karjala & Peter S. Menell, Brief of Amicus Curiae, *Applying Fundamental Copyright Principles to* Lotus Development Corp. v. Borland International, Inc., 10 HIGH TECH. L.J. 177, 179 (1995); Pamela Samuelson, *The Nature of Copyright Analysis for Computer Programs: Copyright Law Professors' Brief Amicus Curiae in* Lotus v. Borland, 16 HASTINGS COMM. & ENT. L.J. 657, 669 (1994).

the language of the CONTU report itself [64] as well as in the scholarly literature.[65] Furthermore, this approach represents the only way to reconcile the apparent inconsistency of the protection envisioned by the CONTU report and the doctrine of *Baker v. Selden*,[66] that copyright protection does not extend to the utilitarian aspects of a literary work. The result reached by the First Circuit is thus correct; to understand why this is so, and why program functionality should be unprotected, we must first consider the *Baker* decision.[67]

The principal import of *Baker* is that utilizing the "art" described in a copyrighted work does not infringe the copyright in the description. Another way of stating this proposition is that an author cannot remove utilitarian material like a system or method from the public domain simply by describing it.[68] The reasoning behind this result is clear. Copyright is designed to reward creativity in expression; because of a general confidence that human language provides for innumerable ways to say essentially the same thing, one needs only the barest showing of originality for copyright protection to attach to expressive material.[69] According to an important policy embodied in the patent law, however, utilitarian art — knowledge concerning a particular way of producing a desired result — is to remain in the public domain unless the inventor can show novelty and nonobviousness.[70] Providing a copyright monopoly in the art or method simply for the (possibly minimal) creative

---

64. *See* CONTU REPORT, *supra* note 4, at 21 ("[O]ne is always free to make the machine do the same thing as it would if it had the copyrighted work placed in it, but only by one's own creative effort rather than by piracy.").

65. *See, e.g.*, Englund, *supra* note 4, at 887, 892, 899.

66. 101 U.S. 99 (1879), as codified in the Copyright Act at 17 U.S.C. § 102(b).

67. *See* Computer Associates v. Altai, Inc., 982 F.2d 693, 704 (2d Cir. 1993) ("The doctrinal starting point in analyses of utilitarian works, is the seminal case of [*Baker*].").

68. *See* Samuelson, *supra* note 63, at 663.

69. *See, e.g.*, Feist Publications, Inc. v. Rural Telephone Serv. Co., 499 U.S. 340, 345 (1991) (citing 1 NIMMER & NIMMER, *supra* note 4, §§ 1.08[C][1], 2.01[A], 2.01[B]).

70. *See, e.g.*, Bonito Boats v. Thunder Craft Boats, 489 U.S. 141, 156 (1989) (holding that state trade secret law was federally preempted because it accorded "patent-like protection to intellectual creations which would otherwise remain unprotected as a matter of federal law"); *see also* DONALD S. CHISUM, 2 PATENTS § 5.01, at 5-11 ("The general purpose behind the requirement of nonobviousness is the same as that behind the requirement of novelty. It serves to limit patent monopolies to those inventions that in fact serve to advance the state of the useful arts."). This point is stressed by Judge Boudin in his concurrence. *See supra* note 49 and accompanying text. As employed here, the term "utilitarian art" essentially means patentable subject matter. Since it was a process, Selden's method was proper subject matter for a patent, though in all probability it failed to meet the other tests of patentability. *See* Englund, *supra* note 4, at 877 n.55.

contribution of the description would thus upset the careful balance struck by the patent regime.[71]

With regard to the functional aspects of copyrighted works, then, § 102(b) is probably best understood as simply incorporating the rule in *Baker* that utilitarian art remains unprotected by a copyright in its description. In all likelihood, Congress was just being thorough in providing a list of terms describing this subject matter, such as "process" and "system." Thus, courts probably should not attach too much significance to the specific meanings of the individual words in the list; in particular, the First Circuit's heavy reliance on the meaning of the term "method of operation" is extremely suspect.[72]

The discussion to this point has essentially restated venerated copyright doctrine; the crux is to apply this theoretical framework to computer programs. At the outset, it is fairly clear that grafting the principles of *Baker* in the obvious way onto programs would yield no copyright protection at all: programs exude functionality in their every aspect.[73] Furthermore, an analysis based on distinguishing electronic computers from physical machines must be doomed, since electrons

---

71. *See* Baker v. Selden, 101 U.S. 99, 102 (1879) (noting that patent-like protection without a showing of novelty would work "a surprise and a fraud upon the public").

72. This reliance is especially unappealing in light of the use of the term in *Baker* itself; there, the Court was talking about mathematics, which has nothing to do with a "method by which one operates something." *See Baker*, 101 U.S. at 103. Most likely, Congress simply copied this somewhat nebulous term into its list in § 102(b).

It is perhaps instructive to compare the function/expression distinction explained here with the traditional idea/expression dichotomy of copyright doctrine. *See, e.g.*, 3 NIMMER & NIMMER, *supra* note 4, § 13.03[A][1], at 13-31 to 13-33. For a true "literary work" in the everyday sense, like a Shakespearean play, every aspect of the work is expressive, even its broadest themes; "idea" is simply the label given to any element above the level of generality where, in a court's view, protection would unduly hamper the original creative freedom of subsequent authors. In this context, the dichotomy is extremely difficult to administer. *See, e.g.*, Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Nobody has ever been able to fix that boundary, and nobody ever can."). The function/expression distinction in the *Baker* context, however, is much easier to apply, since here the notoriously elusive meaning of the mantra "copyright protects only expression" is actually clear: there is nothing expressive about an accounting method. One could choose to label the unprotected art in Selden's book an "idea" as well. *See, e.g.*, 1 NIMMER & NIMMER, *supra* note 4, § 2.18[B], at 2-204. It is probably better, however, to minimize confusion by reserving the term "idea" for non-functional content. *See infra* note 75 and accompanying text. Ultimately, of course, the labels are of little consequence as long as the analysis is done properly. *Cf. Lotus Development Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 822 (1995) (Boudin, J., concurring) ("In all events, the choices are important ones of policy, not linguistics."), *aff'd by an equally divided Court*, 116 S. Ct. 804 (1996).

73. Some commentators have argued for this view. *See, e.g.*, Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663, 741.

running through switches are every bit as functional as cams and gears.[74] For better or worse, however, Congress declared that programs are copyrightable subject matter. The proper question to ask thus becomes the following: What, if anything, about a general purpose machine running a program sets it apart from a non-programmable machine and renders it susceptible of copyright protection? If the answer to this question is "nothing," then applying copyright to computer programs in a convincing way would seem hopeless. However, there is a distinction that is precisely the element on which protection for computer programs should rest.

The key point is that by itself a general purpose computer can do nothing: it requires instructions embodied in a program to generate the functionality which renders it useful by allowing it to perform tasks. The programmer must bend the machine to her control by, in the suggestive parlance of computer science, writing "instructions" to the machine in a "language." Thus, one can quite easily think of a program as a description *to the machine* of a desired functionality, to be performed when the program is executed. Under this view, the program would be analogous to Selden's description of his process. Applying the result in *Baker* to this framework would limit copyright protection to the description of the functionality — the program — in its literal and non-literal elements, while leaving the utilitarian aspect of the work — the program functionality itself — unprotected.[75]

Of course this analogy is flawed, since, as noted above, the program code itself is functional.[76] It makes sense to protect the code itself, however, on the grounds that *Baker*'s policy concerns are much weaker, since the code is generally hidden from the user.[77] As long as there is sufficient variability — multiple ways of writing program code to

---

74. Indeed, a general-purpose computer running a program is a machine for patent purposes. *See, e.g., In re* Alappat, 33 F.3d 1526, 1545 (Fed. Cir. 1994).

75. As used here, "literal" refers to the program code itself, while non-literal elements may include such things as the structure of the code. *See, e.g.*, Computer Associates Int'l., Inc. v. Altai, Inc., 982 F.2d 693, 702-03 (2d Cir. 1992). It may be useful to determine the scope of protection in the non-literal elements of the program code, where everything is "descriptive" in the sense presented here, by applying the usual abstractions test endorsed by the *Altai* court and Professor Nimmer. This abstractions analysis should not, however, extend to program functionality. *Cf. supra* note 38 and accompanying text (discussing the First Circuit's criticism of characterizing the 1-2-3 interface as an abstraction).

76. In fact, each statement in a computer program corresponds directly to a step in the process which occurs inside a computer when the program's instructions are carried out. As explained in the text, however, there are generally multiple processes of this type which generate the same functionality.

77. Indeed, program source code is usually jealously guarded as a trade secret. In the form released to the public, software is generally indecipherable without the application of sophisticated reverse engineering techniques.

implement the same functionality — it will be possible for copyright to inhere separately in multiple machines which are exactly interchangeable to the end users, just as the multiple descriptions of Selden's accounting method were separately copyrightable.[78] By contrast, although there may be many ways to design a user interface which are equivalent, once one becomes popular, the interoperability effects noted by Judge Boudin may skew the market towards a particular choice simply because customers have learned and relied on it. This is the point ignored by Judge Keeton and emphasized by the First Circuit: though it seems clear that there is no merger regarding the choice of possible user interfaces, the existence of choices alone is insufficient for copyrightability in utilitarian works. There must be some affirmative reason to suspend application of the *Baker* rule. Such a reason, which exists for the program code, is absent with regard to program functionality. This functionality should thus remain unprotected.[79]

Admittedly, the above explanation may seem to stretch traditional copyright principles beyond recognition.[80] In particular, extending the concept of expression to embrace human-machine communication may seem distasteful. But this is the only viable explanation for the intuition that CONTU must have relied on in recommending that computer programs be brought into the ambit of copyright law. Furthermore, apart from this understanding, there is no tenable distinction between a computer running a program and any other machine; accordingly, except through the application of extremely fact-specific considerations like those on which Judge Boudin relies, software copyright would make little sense in light of traditional copyright principles.

The explanation presented here is supported by CONTU's description of *Baker*[81] and the distinction between program function and code put forth in the CONTU report itself.[82] Furthermore, treating "the

---

78. If there is no variability, idea and expression "merge," and the expression may be copied without infringement. *See* 3 NIMMER & NIMMER, *supra* note 4, § 13.03[B], at 13-76 to 13-82.

79. *Cf.* 3 NIMMER & NIMMER, *supra* note 4, § 13.03[F], at 13-130 (noting that a process/expression dichotomy may be the proper way to conceive of the respective poles of protected and unprotected material). The problem with the term "process" is that it may be too narrow; it is not clear that every aspect of program functionality is a "process," at least in the patent sense. It is thus probably better to use the more general "functionality," in recognition of the fact that a computer running a program is a machine and can be patented as such.

80. This is probably what Judge Boudin meant by the "cookie cutter" approach. *See supra* text accompanying note 53.

81. *See* Englund, *supra* note 4, at 887.

82. *See id.* at 902 n.169. The statement on this point is misleading: "The movement of electrons through the wires and components of a computer is precisely that process over which copyright has no control." CONTU REPORT, *supra* note 64, at 22. Taken on its face,

program" as fundamentally distinct from functionality when the program is executed appeals to basic intuition. Although the term "program" is often used loosely to encompass functionality in addition to literal code, it seems logical to think of different code implementing the same functionality as a different program. This is, in fact, the understanding in § 101 of the Copyright Act.[83] Therefore, it seems reasonable that copyright should protect only the program itself, and not the "certain result" that the program brings about.[84]

Thus copyright should not extend to Lotus's menu hierarchy; it is just the user interface to the machine created when the computer carries out the instructions embodied in the Lotus 1-2-3 code. This, it seems, is the import of the First Circuit's VCR buttons analogy. A VCR is not copyrightable because it is a machine. It is also, in fact, an electronic computing device, just not a general purpose one (the functionality of the VCR is probably hard-wired, so there is no "code" as such). But now suppose a manufacturer decides to replace its VCR with one identical in all respects save one: its innards have been supplanted by a general-purpose computer and a program encoded into a ROM chip. On Judge Keeton's rationale in the *Borland* District Court decision, the VCR interface would have to be protectable, absent merger, since it was being run by a computer program. This result, in which the copyrightability of an arrangement of buttons depends upon the choice of methods of implementation entirely invisible to the user, illustrates the contradiction inherent in protecting user interfaces for computer programs but not for other utilitarian devices.[85]

---

this statement would mean that nothing in software is protectable, since there is a one-to-one correspondence between sequences of code and movements of electrons. To make sense of this, one must abstract from the "process" of moving electrons to "functionality."

83. "A 'computer program' is a *set of statements or instructions* to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (1994) (emphasis added).

84. *See* Karjala & Menell, *supra* note 63, at 179. The First Circuit's characterization of Borland's activity as "literal copying" of the 1-2-3 interface is thus probably unfortunate. Lotus Development Corp. v. Borland Int'l, Inc., 49 F.3d 807, 814 (1st Cir. 1995), *aff'd by an equally divided Court*, 116 S. Ct. 804 (1996). This terminology is potentially very misleading, since according to customary usage, "literal" addresses the core of what copyright protects — in this case, the words and symbols of the program code. It is difficult to see how the term "literal copying" can apply any more to the unprotected utilitarian element in *Borland* than to Selden's method.

85. Note that the First Circuit seems to get the analysis here somewhat backwards, and thus misses the full strength of its own argument. The arrangement of buttons would be unprotected, says the court, even if copyright inhered in our VCR, because the buttons are a "method of operation." *Borland*, 49 F.3d at 817. But the point seems to be that, in light of the important policy reasons why copyright does not protect utilitarian devices like VCRs, to encroach upon this principle simply because a computer program lurks within the machine makes scant sense. *See also* Synercom Tech., Inc. v. University Computing Co.,

Consider another example. The human-machine interface of an airplane cockpit is protectable only by patent, if at all. But suppose a company creates a cockpit design which, though original, is unpatented. Instead of building it into an airplane, however, the company first releases the design as a computer simulation. Under Judge Keeton's analysis, another company would be free to put the cockpit design in its own plane, but could not write its own simulation program. It is difficult to see any viability in the distinction between an interface and a simulated interface for purposes of protecting the creativity of the design.

The concerns Judge Keeton raises in *Paperback* and *Borland* regarding the piracy of program functionality are valid ones, but they are adequately addressed by the patent law. It has traditionally been the province of patent law to protect utilitarian creativity.[86] Furthermore, it is especially compelling that protection be left to the patent law in the interface context, since the Patent and Trademark Office ("PTO") can address the interoperability concerns noted by the First Circuit in deciding whether the interface embodies a sufficient level of novelty to warrant protection.[87] Standards promulgated by the PTO would certainly allow for higher predictability and lower administrative costs than would a case-specific fair use test administered by the courts. Additionally, these standards would provide notice to would-be copiers before the putatively infringing copy has been developed and released. Judge Keeton's concern in *Paperback* and *Borland*, that designing a successful interface requires far more creativity than it takes to encode one, is just *Baker* redux. Presumably, no one bought Selden's book for the elegance of the author's prose. But even though the utilitarian "art" — the accounting system — was the point of the book, copyright inhered only in the descriptive elements contributed by the author. Copyright protection limited to the method of encoding a particular function will still protect the significant effort which goes into writing and debugging computer code.

---

462 F. Supp. 1003, 1013-14 (N.D. Tex. 1978) (likening the input formats of a computer program to the "figure-H" pattern of an automobile stick shift and holding the formats uncopyrightable).

86. Englund, *supra* note 4, at 893. Patent protection would inhere in the machine created by a general-purpose computer together with a program. *See supra* note 74.

87. The PTO has already issued patents on software user interfaces. *See* David Bender, *A 1995 Perspective on Software Patents: Part II*, J. PROPRIETARY RTS., Mar. 1995, at 9.

V. CONCLUSION

It is understandable that the First Circuit did not embrace the conceptual view presented here or anything like it. It is hard for any court confronted with the problem of software copyright to disentangle its analysis from confusing and often contradictory approaches that the incredible opaqueness of the statute has engendered in the courts. The First Circuit undoubtedly wanted to narrow the question presented and minimize controversy to the extent possible. In light of these considerations, it may be time for Congress to step in and legislatively mandate the approach outlined in this Note, which is, after all, only an attempt to explain what CONTU seems to have envisioned.