FUNCTIONALITY AND EXPRESSION IN COMPUTER PROGRAMS: REFINING THE TESTS FOR SOFTWARE COPYRIGHT INFRINGEMENT

Pamela Samuelson[†]

ABSTRACT

Courts have struggled for decades to develop a test for judging infringement claims in software copyright cases that distinguishes between program expression that copyright law protects and program functionality for which copyright protection is unavailable. The case law thus far has adopted four main approaches to judging copyright infringement claims in software cases. One, now mostly discredited, test would treat all structure, sequence, and organization (SSO) of programs as protectable expression unless there is only one way to perform a program function. A second, now widely applied, three-step test calls for creating a hierarchy of abstractions for an allegedly infringed program, filtering unprotectable elements, and comparing the protectable expression of the allegedly infringed program with the expression in the second program that is the basis of the infringement claim. A third approach has focused on whether the allegedly infringing elements are program processes or methods of operation that lie outside the scope of protection available from copyright law. A fourth approach has concentrated on whether the allegedly infringing elements of a program are instances in which ideas or functions have merged with program expression. This Article offers both praise and criticism of the approaches taken thus far to judging software copyright infringement and proposes an alternative unified test for infringement that is consistent with traditional principles of copyright law and that will promote healthy competition and ongoing innovation in the software industry.

DOI: https://dx.doi.org/10.15779/Z38WW76Z83

† Richard M. Sherman Distinguished Professor of Law, Berkeley Law School. I am very grateful to Kathryn Hashimoto for excellent research for and editing of this article. I am also grateful to Clark Asay, Jonathan Band, Joshua Bloch, Oren Bracha, Dan Burk, Julie Cohen, Joe Craig, Charles Duan, Shubha Ghosh, Ariel Katz, Peter Lee, Mark Lemley, Glynn Lunney, Corynne McSherry, Christina Mulligan, Aaron Perzanowski, Michael Risch, Christopher Jon Sprigman, Fred von Lohmann, and Phil Weiser for comments on an earlier draft of this article. I also wish to thank Lionel Bently for the opportunity to give the 10th Annual International IP Lecture at Emanuel College at Cambridge University on which this Article was initially based.

TABLE OF CONTENTS

I. INTRODUCTION

The paradigmatic roles of copyright and patent laws have been, respectively, to protect original authorial expressions from illicit copying, and to protect novel and nonobvious functional designs (if they have been appropriately claimed and examined by patent officials) from illicit uses.[1] It would be convenient if copyright law could be assigned the role of protecting the expression in computer programs and patent law the role of protecting program functionality. While courts continue to try to distinguish between program expression and program functionality, this distinction has proven elusive in the decades since Congress decided to extend copyright protection to computer programs.[2]

1. *See* J.H. Reichman, *Legal Hybrids Between the Patent and Copyright Paradigms*, 94 COLUM. L. REV. 2432, 2448–53 (1994) (describing the classical patent and copyright paradigms in the international intellectual property system).

2. While some claim that Congress extended copyright protection to computer programs when it enacted the Copyright Act of 1976, there is some ambiguity in the legislative history on this point. *Compare* FINAL REPORT OF THE NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS 15–16 (1978) [hereinafter CONTU REPORT] (concluding that Congress had extended copyright protection to software in 1976) *with* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663, 694–96 (1984) (suggesting that § 117 in the 1976 Act preserved the status quo of unprotectability under

In the years preceding the enactment of the Copyright Act of 1976,[3] members of Congress were warned that the functionality of computer programs would make it difficult to fit them into the copyright realm.[4] However, lingering concerns about the potential misfit were for a time allayed by a 1978 National Commission on New Technological Uses of Copyrighted Works (CONTU) report that endorsed copyright protection for programs.[5] CONTU observed that "the distinction between copyrightable computer programs and uncopyrightable processes or methods of operation does not always seem to 'shimmer with clarity,'" but it was nevertheless "important that the distinction between programs and processes be made clear."[6] The report expressed optimism that traditional principles of copyright law, when applied to programs, would strike the right balance,[7] and it was content to leave the difficult (and perhaps "futile") task of

---

the prior act as to computer-related subject matters). However, any such ambiguity was resolved by 1980 amendments to the Copyright Act of 1976 (1976 Act), which implemented legislative changes that CONTU recommended in its report. *See* Pub. L. No. 96-517, 94 Stat. 3015 (codified at 17 U.S.C. §§ 101, 117 (1980)).

3. Pub. L. No. 94-553, 90 Stat. 2541 (1976), codified at 17 U.S.C. § 101, et seq.

4. *See* Hearings Before the Subcomm. on Patents, Trademarks, and Copyrights of the S. Comm. on the Judiciary Pursuant to S. Res. 37 on S. 597, 90th Cong. 192–97 (1967), *reprinted in* 9 OMNIBUS COPYRIGHT REVISION LEGISLATIVE HISTORY 192–97 (George S. Grossman ed., 1976) (testimony of Professor Arthur Miller). Miller expressed concern that courts might construe copyright protection for programs as "extend[ing] to or embody[ing] the process, scheme, or plan that the program uses to achieve a functional goal," saying this would confer "patent like protection under the guise of copyright." *Id.* at 197. Congress responded to these concerns by adopting a provision stating that "[i]n no case does copyright protection for an original work of authorship extend to any . . . procedure, process, system [or] method of operation . . . regardless of the form in which it is . . . embodied in such work." 17 U.S.C. § 102(b) (2012). This provision is discussed at length *infra* Part III.

5. CONTU REPORT, *supra* note 2, at 1–2. CONTU acknowledged that there was not "universal agreement" about copyright protection for software. *Id*. at 20–21. *See also* Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 344–46 (1970) (questioning the economic case for extending copyright protection to computer programs). While CONTU was deliberating about new technology issues, the World Intellectual Property Organization was considering a sui generis form of intellectual property protection for software. *See* WORLD INTELLECTUAL PROP. ORG., INT'L BUREAU, MODEL PROVISIONS ON THE PROTECTION OF COMPUTER SOFTWARE (1978). Whether computer programs should be protected by copyright law was not part of CONTU's original charter, which mainly focused on photocopying and digitizing published texts. *See* Samuelson, *CONTU Revisited*, *supra* note 2, at 663 n.2, 699. That may explain why none of the Commissioners had any expertise about computers or computer programs. *Id*. at 699.

6. CONTU REPORT, *supra* note 2, at 18.

7. *See id*. at 12–23. CONTU thought copyright should grant no more economic power than was needed to create proper incentives to create software. *Id*. at 12.

drawing boundaries between program expression and functionality to the judiciary.[8] Unfortunately, CONTU failed to fully understand the intrinsic functionality of computer programs, the importance of standards and network effects in the software industry, and the inherent need to develop software capable of interoperating with other programs. It also failed to offer guidance on how, when, and why functionality should constrain the scope of copyright protection in programs.[9]

Commentators have debated for decades how much legal protection software developers should get from copyright law in order to induce optimal levels of investment in the development of computer programs.[10] Some have worried that copyright protection for programs might either be too "weak" if infringement could be easily avoided by rewriting the same

8. *Id*. at 22–23.

9. According to CONTU, programs were no more functional than sound recordings, *id*. at 10, which was simply not true. After all, the inherent purpose of computer programs is to automate functional processes, whereas the purpose of sound recordings is to allow users to listen to music. *See id*. at 27–29 (Hersey dissent distinguishing program functionality from other copyrighted works). CONTU also asserted that utility had never been a bar to the copyrightability of a work or a limit on the scope of protection available to protected works, *id*. at 19–21, which was also untrue. *See* Samuelson, *CONTU Revisited*, *supra* note 2, at 732–39 (explaining reasons why utilitarian works have conventionally been excluded from copyright protection); *see also* Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329, 1359–61 (1987) (offering other reasons why copyright protection is inappropriate for operating system software).

10. From the late 1980s to the late 1990s, scholars produced an extensive literature about copyright protection for computer programs, particularly their nonliteral elements. *See, e.g.*, Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559 (1994); Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33 (1987); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989); Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977 (1993); J.H. Reichman, *Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research*, 42 VAND. L. REV. 639 (1989); Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994); Lloyd L. Weinreb, *Copyright for Functional Expression*, 111 HARV. L. REV. 1150 (1998); 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 (1990). Especially influential was a law review article, later incorporated into the Nimmer treatise. *See* David Nimmer et al., *A Structured Approach to Analyzing Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625 (1988); MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03 [F](2015) [hereinafter NIMMER ON COPYRIGHT]; s*ee also* JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL (1995); JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL 2.0 (2011).

program in different source code, or too "strong" if programmers felt compelled to do things differently than an existing program in order to avoid infringement, thereby impeding beneficial standardization.[11] That concern has manifested itself in the software copyright cases that followed.

Appellate courts have taken four main approaches to distinguishing the original expression in computer programs from program functionality. A first-in-time, but now much discredited, approach was adopted by the Third Circuit Court of Appeals in *Whelan Associates v. Jaslow Dental Lab., Inc.*, under which the "structure, sequence, and organization" (SSO) of computer programs was deemed protectable expression unless there was only one way to perform a function (in which case a second comer could use the same SSO under the merger of idea and expression doctrine).[12]

A second was the Second Circuit Court of Appeals' approach in *Computer Associates Int'l, Inc. v. Altai, Inc.*[13] The *Altai* decision was highly critical of *Whelan* and its test for software copyright infringement.[14] As an alternative, *Altai* offered the abstraction-filtration-comparison (AFC) test for judging copyright infringement claims in computer program cases.[15] *Altai*'s principal contribution has been its insistence that courts must "filter" out unprotectable elements of programs, such as those necessary for achieving interoperability with other programs, before assessing infringement claims.[16] The AFC test has been adopted and applied in numerous subsequent cases.[17]

---

11. *See, e.g.*, Breyer, *supra* note 5, at 347–48; *see also* Samuelson et al., *Manifesto*, *supra* note 10, at 2356–63 (explaining why applying copyright law to computer programs might lead to cycles of under- and over-protection).

12. 797 F.2d 1222, 1236, 1248 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987). *Whelan* is discussed *infra* text accompanying notes 53–70.

13. 982 F.2d 693 (2d Cir. 1992).

14. *Altai*'s criticism of *Whelan* is discussed *infra* text accompanying notes 82–85.

15. *Altai*, 982 F.2d at 706–11.

16. *Id*. at 707–10.

17. The *Altai* approach to software copyright infringement has been endorsed in numerous other circuit court decisions. *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524–25 (9th Cir. 1992); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 834 (10th Cir. 1993); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1543–44 (11th Cir. 1996). As of July 6, 2015, *Altai* had been positively cited in 389 federal court decisions on the subject of copyright in the Lexis database and had been cited in all circuits. *See also* Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 BERKELEY TECH. L.J. 1, 14–15 (1995) (treating *Altai* as the leading case on copyright protection for computer programs).

A third approach was used by the First Circuit Court of Appeals in *Lotus Dev. Corp. v. Borland Int'l, Inc.*[18] *Borland* ruled that the command hierarchy of a spreadsheet program was an integral part of a method of operation that 17 U.S.C. § 102(b) excluded from the scope of copyright protection in programs.[19] The court compared the Lotus command structure to the clearly uncopyrightable set of buttons that control the operation of videotape machines.[20]

A fourth approach emerged in the Sixth Circuit Court of Appeals' decision in *Lexmark Int'l, Inc. v. Static Control Components*.[21] The court decided that a computer program embedded in Lexmark's printer cartridges that competitors had to install to enable cartridges to interoperate with Lexmark printers was ineligible for copyright protection.[22] Because the idea or function of the program and its expression had merged, the program was held to be uncopyrightable.[23]

A common thread through the *Altai*, *Borland*, and *Lexmark* decisions is that copyright infringement does not occur when a second comer must copy some aspects of another firm's program to achieve compatibility. Courts have deemed the functional requirements for achieving compatibility to be unprotectable elements of these copyrighted programs, even though more than a modicum of creativity may have imparted originality to these elements.

The seeming consensus that program interfaces necessary for interoperability are unprotectable by copyright law was recently called into question by the Court of Appeals for the Federal Circuit (CAFC) in *Oracle Am., Inc. v. Google Inc.*[24] At issue was whether the command structure of certain elements of the Java application program interface (API) was protectable by copyright law. The CAFC reversed a lower court ruling that this command structure was an unprotectable method of operation, or

18. 49 F.3d 807, 814–15 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996)

19. *Id.* at 815–18. That section provides that "[i]n no case does copyright protection for an original work of authorship extend to any . . . procedure, process, system [or] method of operation . . . regardless of the form in which it is . . . embodied in such work." 17 U.S.C. § 102(b). The *Borland* decision is discussed at length in Section III.B.

20. *Borland*, 49 F.3d at 817.

21. 387 F.3d 522 (6th Cir. 2004).

22. *Id.* at 542–43.

23. *Id.* at 541–42. The *Lexmark* decision is discussed in Section IV.D.

24. 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015). The CAFC's *Oracle* decision is discussed at length in Sections III.C & D, Section IV.E, and Section V.B.

alternatively that copyright protection was unavailable under the merger doctrine.[25]

Although the CAFC in *Oracle* purported to defer to *Altai*,[26] the ghost of the discredited *Whelan* decision reappeared in the CAFC's endorsement of the copyrightability of program SSO.[27] Like the Third Circuit in *Whelan*, the CAFC in *Oracle* viewed § 102(b) as merely a restatement of the idea/expression distinction and treated the merger doctrine as applicable only when there was, ex ante, no other way for engineers to design this or other program SSO.[28] The CAFC was untroubled by the prospect that software developers might obtain both patent and copyright protection for APIs of computer programs.[29] There was, in its view, no need to sort out functionality and expression in computer programs. Copyright could protect both as long as there was a modicum of creativity to support the claim of copyright.

The *Oracle* decision has rekindled a decades-old debate, which many thought had been settled in the late 1990s, about the proper scope of copyright protection for computer programs and how courts should analyze claims of software copyright infringement.[30] The Supreme Court decision not to review *Oracle* leaves the CAFC ruling intact for the time being.[31]

---

25. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 977, 998–1002 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

26. *Oracle*, 750 F.3d at 1355–58.

27. *See id*. at 1366–68.

28. *See id*. at 1359–62 (discussing the merger doctrine), 1364–68 (discussing § 102(b)).

29. *See id*. at 1380–81.

30. The *Oracle* decision has already spawned new rounds of software copyright litigation. *See, e.g*., Cisco Sys., Inc. v. Arista Networks, Inc., No. 5:14-cv-05344 (N.D. Cal. 2014); Synopsys, Inc. v. ATopTech, Inc., No. 3:13-cv-02965 (N.D. Cal. 2013). The complaints in both cases include patent as well as copyright infringement claims. *See* Complaint for Copyright and Patent Infringement, Cisco Sys., Inc. v. Arista Networks, Inc., No. 14-5344, 2014 WL 6844640 (N.D. Cal. Dec. 5, 2014); Amended Complaint for Copyright Infringement, Patent Infringement, Breach of Contract, and Breach of Implied Covenant of Good Faith and Fair Dealing, Synopsys, Inc. v. ATopTech, Inc., No. 3:13-cv-02965-MMC, 2013 WL 7117632 (N.D. Cal. Nov. 25, 2013). Appeals from the District Court decisions in *Arista* and *Synopsys* will go to the CAFC because the plaintiffs in those cases have learned a lesson from the *Oracle* case that tacking on a patent claim will avoid going to the Ninth Circuit where compatibility and § 102(b) defenses would be more likely to prevail.

31. 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015). The CAFC's ruling will not be binding on the Ninth Circuit or other courts. The *Oracle* case was remanded for retrial of Google's fair use defense, resulting in a jury verdict in favor of Google and fair use. Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA, 2016 WL 3181206 (N.D. Cal. June 8, 2016), *appeal docketed*, Nos. 17-1118, -1202 (Fed. Cir. Nov. 14, 2016).

This Article aims to provide guidance about how courts should assess claims of copyright infringement in computer program cases. It assesses the strengths and limitations of the various tests for infringement adopted in software copyright cases and offers a refined test for infringement that takes the soundest features from the existing tests and consolidates them into one unified approach.

Part II reviews the *Whelan* and *Altai* decisions and explains why the AFC test is more consistent with traditional principles of copyright law than the *Whelan* is-there-any-other-way-to-do-it test. *Altai* recognized that external factors, such as the need to be compatible with other programs, sometimes constrain the design decisions of subsequent programmers, and when this happens, those constraints limit the scope of copyright protection in programs. While there is much in the decision to praise, *Altai* failed to heed the statutory directive in § 102(b) that procedures, processes, systems, and methods of operation should also be filtered out before making judgments on copyright infringement claims in software cases.

Part III explains the important role that § 102(b) has played in various computer program cases, including *Borland*. It then discusses the numerous respects in which the CAFC in *Oracle* misinterpreted § 102(b). It considers six types of cases in which courts have held that aspects of programs that are necessary for achieving interoperability with other programs or hardware are too functional to be protected by copyrights.

Part IV explains why the merger doctrine has an important role to play in the assessment of infringement claims involving computer programs and why the CAFC erred in its interpretation of this doctrine. Courts should explicitly recognize a merger of function and expression doctrine in some computer program cases. That doctrine would usefully complement an analysis of elements that may be unprotectable under § 102(b).

Part V considers the roles that copyright and patent law should play in protecting program innovations, with particular attention to how courts should assess claims that copyright protection should be unavailable to aspects of programs possibly eligible for patent protection. The CAFC in *Oracle* conflated copyright and utility patent protections for software, as though it was unnecessary to even try to distinguish program expression and functionality.

Part VI offers a pragmatic approach to distinguishing between program functionality and expression in copyright cases, as well as a refined version of the *Altai* AFC test that is consistent with the traditional principles of copyright law and the overwhelming majority of software copyright cases (even if not consistent with the CAFC's *Oracle* decision). Competition and ongoing innovation will better thrive when the scope of copyright protection

is relatively thin, allowing programmers to reuse functional design elements and know-how that will promote the progress of science and useful arts, as the Constitution directs.[32]

II. THE ABSTRACTION-FILTRATION-COMPARISON TEST

The Second Circuit's 1992 *Altai* decision was important for a number of reasons. For one thing, *Altai* recognized that the essentially utilitarian nature of computer programs meant that copyright law should be applied carefully to ensure that courts were not extending protection to functional aspects of programs, which should be free for all programmers to use (unless patented).[33] Second, *Altai* recognized that software developers are often constrained in their design decisions by, among other things, the need to be compatible with existing software or hardware, which should limit the scope of copyright protection in programs.[34] Third, the Second Circuit indicated in *Altai* that SSO was not a fruitful concept to employ when analyzing whether nonliteral elements of programs were within the scope of copyright protection.[35] Fourth, *Altai* rejected sweat-of-the-brow arguments for giving software a broad scope of protection because such arguments were inconsistent with Supreme Court precedent and fundamental principles of copyright law.[36] If software developers needed more legal protection than copyright could provide, the Second Circuit thought this was a matter for Congress.[37]

Fifth and most important for the purposes of this section, *Altai* provided courts in subsequent cases with a more nuanced test for judging claims of software copyright infringement than *Whelan* and other prior cases had provided. The first step of *Altai*'s so-called "abstraction-filtration-comparison" test was to create a hierarchy of abstractions for the program alleged to be infringed, the second was to filter out unprotectable elements, and the third was to compare the remaining expression in the plaintiff's program with the defendant's program to determine whether the defendant infringed copyright.[38]

In each respect, the *Altai* decision countered approaches that courts had taken in earlier cases. Section A discusses the cases to which *Altai* was, in

32. *See* U.S. CONST. art. I, § 8, cl. 8.
33. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 703–05 (2d Cir. 1992).
34. *Id.* at 707–10.
35. *Id*. at 706.
36. *Id*. at 711–12 (citing Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 349–50 (1991)).
37. *Id*. at 712.
38. *See id* at 706–11.

a sense, responding. Section B provides some detail about the AFC test and why the Second Circuit thought many elements of programs should be filtered out before making final judgments about infringement. Section C explains what *Altai* got right and which aspects of the AFC test are in need of some refinement.

A. THE ROCKY ROAD TO *ALTAI*

In the early 1980s, U.S. appellate courts reviewed two software copyright cases in which the defendants had made exact copies of computer program object code.[39] Both defendants argued that the exact copying of the Apple II operating system (OS) programs was necessary to enable their computers to be compatible with the Apple II so that programs written to run on the Apple II could run on the defendants' platforms as well.[40] Apple prevailed in both cases.

Of the two cases, the Third Circuit's decision in *Apple Computer, Inc. v. Franklin Computer Corp.* merits discussion because of its unnuanced responses to the defendant's merger and § 102(b) arguments.[41] Franklin contended that the ideas of the Apple II OS had merged with their expression because the only way Franklin could make its computers functionally compatible with the Apple II was by installing exact copies of the Apple OS programs on its machines.[42] The Third Circuit rejected this argument, stating that achieving compatibility was "a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[43] Franklin also contended that the Apple OS programs were unprotectable by copyright law because § 102(b) excluded functional processes from the scope of copyright protection.[44] Section 102(b) provides that "[i]n no case does copyright protection . . . extend to any idea, procedure, process, system, method of operation, concept, principle or discovery, regardless of the form in which it is . . . embodied in . . . [the] work."[45] But the court

39. Apple Comput., Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240 (3d Cir. 1983).

40. Formula's compatibility claim was more indirect than Franklin's, but Formula, like Franklin, was selling clones of the Apple II computer, and Formula objected to the issuance of an injunction because it would inhibit competitive entry into the computer market. *Formula*, 725 F.2d at 522–26; *Franklin*, 714 F.2d at 1253.

41. *Franklin*, 714 F.2d 1240.

42. *Id.* at 1253.

43. *Id.*

44. *Id*. at 1250–52.

45. 17 U.S.C. § 102(b).

rejected Franklin's argument because Congress had decided to treat programs as literary works.[46]

The court in *Franklin* unquestionably reached the right result. If Congress' decision to extend copyright protection to computer programs was to be respected, it had to mean that exact copying of object code would get defendants in trouble, particularly where, as in *Franklin*, the copyist had not even tried to reimplement the Apple II OS functionality in different code.[47]

The most troubling aspect of the *Franklin* decision was its strident rejection of compatibility as a possible justification for some copying from an existing program. Although eventually repudiated in several subsequent cases,[48] this dictum was recently revived in the CAFC's *Oracle* decision.[49] *Franklin* also took an unduly narrow view of § 102(b). The Third Circuit was right that § 102(b) should generally not be interpreted to allow the exact copying of object code just because such code is a "process."[50] But the court failed to acknowledge that § 102(b) excluded more than abstract ideas from the scope of copyright protection.[51] The statutory exclusion of methods and processes embodied in programs had to be respected as well. Because of this exclusion, the scope of copyright protection in computer programs should be thinner than the scope of protection available to conventionally expressive works, such as novels and plays.[52]

Although literal copying of program code, as in *Franklin*, presented an easy question for courts to answer, a more difficult question has been whether nonliteral elements of programs, such as SSO, qualify for copyright protection. That question was first addressed at the appellate level by the Third Circuit in *Whelan Associates, Inc. v. Jaslow Dental Lab., Inc.*[53]

---

46. *See Franklin*, 714 F.2d at 1248–49, 1252–53.

47. *Id.* at 1245.

48. *See infra* text accompanying notes 259–264, 363.

49. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014) (quoting the *Franklin* dicta).

50. *Franklin*, 714 F.2d at 1250–52. In rare cases, such as *Lexmark*, function and expression in a program may merge. Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 540-42 (6th Cir. 2004). *See infra* text accompanying notes 318–322 for a discussion of function/expression merger.

51. *See* 17 U.S.C. § 102(b) ("In no case does copyright protection . . . extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery . . . .").

52. Congress added § 102(b) to the 1976 Act in part to ensure that courts would not give too broad an interpretation to software. *See* H.R. REP. NO. 94-1476 at 56–57 (1976), *as reprinted in* 1976 U.S.C.C.A.N. 5659, 5670.

53. 797 F.2d 1222 (3d Cir. 1986). A few District Court decisions prior to *Whelan* treated structural elements of programs as protectable expression. *See, e.g.*, SAS Inst., Inc.

Jaslow had commissioned Whelan to develop a program to automate common business processes of dental laboratories.[54] The parties initially intended to exploit the software as partners,[55] but after a falling out, Jaslow decided to develop a program to do the same functions that would run on IBM-PCs.[56] Although Jaslow's and Whelan's programs were written in different programming languages and used different algorithms and data structures, the appellate court was impressed by similarities in the overall structures of the two programs,[57] in their file structures,[58] and in the way five modules performed functions.[59] Based on these similarities, the appellate court upheld a ruling that Jaslow infringed Whelan's copyright.[60]

Jaslow's main defense was his contention that Congress had intended for copyright law to protect programs only against exact code copying.[61]

---

v. S&H Comput. Sys., 605 F. Supp. 816 (M.D. Tenn. 1985). *Whelan* relied in part on the *SAS* case. *See* 797 F.2d at 1239.

54. Whelan Assoc., Inc. v. Jaslow Dental Lab., Inc., 609 F. Supp. 1307, 1309–10 (E.D. Pa. 1985). Jaslow had tried to develop this program on his own, but he lacked the skills to complete it, which is why he hired Whelan. *Id*.

55. *Id*. at 1312. Jaslow terminated this agreement and started his own firm to sell a competing product. *Id*. at 1313–15.

56. Whelan's contract explicitly provided that she would own IP rights in the Dentalab software. *Id*. at 1310. Despite this, Jaslow claimed to be the sole owner of the program (or at least a co-author of it) and licensed that software to third parties. *Id*. at 1315–17. The court held him as an infringer for the revenues he collected for selling Whelan's program to others. *Id*. at 1323.

57. 797 F.2d at 1239, 1247–48. In addition to selling infringing copies of Whelan's program, Jaslow, with the help of a contract programmer, had developed a competing program, which Whelan claimed infringed her copyright. 609 F. Supp. at 1314–15. This was the program that Whelan alleged had non-literally infringed her copyright. He called the new program "Dentlab," which the court held was confusingly similar to Whelan's "Dentalab" trademark. *Id*. at 1324.

58. *Whelan,* 797 F.2d at 1242–43.

59. *Id*. at 1245–46. The District Court had a difficult time understanding the technology at issue. Judge Vanartsdalen decided that Whelan's expert was more credible than Jaslow's expert because the former had observed the program in operation whereas the latter had only studied the source code for the two programs. *Whelan*, 609 F. Supp. at 1316. Vanartsdalen was also particularly impressed by the similarities in screen displays, *id*. at 1322, even though Whelan's claim of infringement was based on copying of structure from the underlying text of the program, not on screen displays. The District Court noted that Jaslow's program was not a translation from one computer language to another. *Id*. at 1315. The appellate court, on the other hand, was more precise about the structural similarities and recognized the risk of undue prejudice if courts used similarities in screen displays as evidence of similarities in program text, given that independently written code can produce the same results. 797 F.2d at 1244–45.

60. *Id*. at 1240–42.

61. *Id*. at 1235, 1241–42. *See also* Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 U. DAYTON L. REV. 975, 984–89 (1994) (arguing that copyright should protect only against slavish copying of

The Third Circuit rejected this argument, reasoning that because copyright law had long protected structural elements of conventional literary works, such as novels and plays,[62] copyright should protect the SSO of programs as well.[63] SSO included, in its view, "the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information."[64] Just as with novels and plays, anyone was free to copy the ideas from existing programs, but program SSO, *Whelan* announced, was protectable expression as long as there was more than one way to structure a program to achieve the program's functions.[65] Because Jaslow could have used different SSO, his use of the same or similar ones as Whelan's constituted infringement.[66]

The Third Circuit acknowledged that software was a utilitarian work,[67] but it made no effort to distinguish between nonliteral elements of programs that should be regarded as protectable structures and those that might be unprotectable utilitarian processes.[68] *Whelan*'s test for infringement effectively rendered all program structure as protectable subject matter (unless there was truly no other way to structure the program). The Third Circuit noted that program structure and logic were "among the more significant costs in computer programming."[69] The court concluded that without broad copyright protection for computer programs, there would be too little legal protection to provide proper incentives to invest in developing computer programs.[70]

---

code). Jaslow infringed when he sold Whelan's program, *see supra* note 56, but in light of later case developments, he may not have infringed as to the later-developed program. *See infra* note 450.

62. *See, e.g.*, Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936) (finding infringement because of structural similarities between scenes in plaintiff's play and defendant's movie).

63. *Whelan*, 797 F.2d at 1233–34.

64. *Id*. at 1239–40 (quoting *Whelan*, 609 F. Supp. at 1320). This would seem to extend copyright protection to all program behavior, which is generally highly functional in character.

65. *Id*. at 1236; *see also id*. at 1224 n.1 (indicating that the opinion uses "'structure,' 'sequence,' and 'organization' interchangeably when referring to computer programs").

66. *Whelan*, 797 F.2d at 1242–48.

67. *Id*. at 1236.

68. CONTU did not provide guidance on this score either. *See supra* text accompanying note 8.

69. *Whelan*, 797 F.2d at 1237.

70. *Id*. The Second Circuit in *Altai* noted that *Whelan* had been decided prior to the Supreme Court's revocation of the sweat-of-the-brow doctrine in Feist Publ'ns, Inc. v.

Whelan and its test for infringement were initially followed in some cases,[71] despite sharp criticisms in the law review literature for taking an overly expansive view of the scope of protection for programs.[72] The *Whelan* test posited that only the general purpose or function of a program was an unprotectable idea under § 102(b), and all program SSO, no matter how abstract or standard it might be in the programming field, was protectable expression unless there was truly no alternative way to accomplish the function.[73] The first appellate court to break with *Whelan* was the Fifth Circuit Court of Appeals in *Plains Cotton Cooperative Ass'n v. Goodpasture Computer Service, Inc.*[74] The Fifth Circuit refused to follow *Whelan* because many of the structural similarities between user interfaces of the cotton trading programs at issue were "dictated by the externalities of the cotton market" and to be expected of a marketable program in that field.[75]

Although *Whelan* did not involve program SSO necessary for achieving interoperability with other programs, coupled with *Franklin*'s rejection of compatibility defenses, the court's broad endorsement of SSO as protectable expression put unlicensed developers who reimplemented the interfaces of an existing program to make their own programs interoperable at risk of infringement.[76] Relying on the *Whelan* decision's endorsement of copyright protection for program SSO, Computer Associates (CA) brought a lawsuit against Altai for nonliteral copyright infringement that came before the Second Circuit in 1992.[77] Altai defended by arguing that it was

---

Rural Tel. Serv. Co., 499 U.S. 340, 349–50 (1991). Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 711–12 (2d Cir. 1992).

71. *See*, *e.g.*, Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986).

72. *See Altai*, 982 F.2d at 711–12 (citing five critics of the *Whelan* decision); Englund, *supra* note 10, at 875–76 (noting that not just program ideas, but also program processes, should be excluded from the scope of copyright protection). See also *infra* notes 82–85 and accompanying text for a fuller account of *Altai*'s criticism of *Whelan*.

73. *Whelan*, 797 F.2d at 1236.

74. 807 F.2d 1256 (5th Cir. 1987).

75. *Id*. at 1262.

76. One post-*Whelan* District Court upheld a second-comer's reuse of a first-comer's variation on an interface protocol that was necessary to achieve compatibility. *See* Secure Servs., Inc. v. Time & Space Proc'g, 722 F. Supp. 1354 (E.D. Va. 1989) (finding no infringement to re-implement digital handshake for secure fax machines to be sold to the government because variations lacked sufficient originality). *Whelan* was not cited.

77. *See* Reply Brief for Plaintiff-Appellant at 8–18, Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992) (No. 91-7893), 1991 WL 11010234 (relying heavily on *Whelan*).

necessary to use the same SSO in its program in order to be compatible with third-party programs.[78]

B. THE *ALTAI* DECISION AND THE AFC TEST

Computer Associates Int'l, Inc. v. Altai, Inc. was the first appellate court decision to consider a compatibility defense to a claim of nonliteral copyright infringement.[79] CA charged Altai with copyright infringement for copying program SSO—overall structure, macros, lists of services, and parameter lists (i.e., lists of information that needed to be sent and received by subroutines of the affected programs)—from CA's scheduling program.[80] Altai argued that it needed to use the same parameter lists and macros because this SSO was necessary for its programs to be compatible with the IBM computers on which both its own and CA's scheduling programs were designed to run; other similarities were, moreover, to be expected in programs of that kind.[81]

Before setting forth its alternative analysis, the Second Circuit discussed the *Whelan* decision at some length. The *Altai* court agreed with the Third Circuit that nonliteral elements of programs could be protected by copyright,[82] but it did not find *Whelan*'s SSO concept helpful in distinguishing between which nonliteral elements of programs were protectable by copyright law and which were not. The court regarded the SSO concept as "demonstrat[ing] a flawed understanding of a computer program's method of operation" and as resting on a "somewhat outdated appreciation of computer science."[83] While it characterized *Whelan* as "the most thoughtful" attempt to apply the idea/expression distinction to computer programs, the court also noted the widespread criticism of the decision as "conceptually overbroad."[84] The *Whelan* test for infringement of program SSO was, moreover, "descriptively inadequate," relying "too

78. Brief of Defendant-Appellee at 10–13, Comput. Assoc. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992) (No. 91-7893), 1991 WL 11010233.

79. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 775 F. Supp. 544 (E.D.N.Y. 1991), *aff'd*, 982 F.2d 693 (2d Cir. 1992).

80. CA sued Altai for copyright infringement after learning that Altai's program contained code directly copied from its CA-Scheduler program by one of CA's former employees whom Altai had hired. Altai purged the tainted code from its program and assigned a clean-room team of programmers to reimplement the compatibility components in new non-infringing code. But CA argued that there were still substantial similarities in SSO that Altai had copied from its program. *Altai*, 982 F.2d at 697–700.

81. *See id.* at 714–15.

82. *Id*. at 702.

83. *Id*. at 706.

84. *Id*. at 705.

heavily on metaphysical distinctions" instead of on "practical considerations."[85]

The "essentially utilitarian nature of a computer program," the court said in *Altai*, makes distinguishing program ideas and expressions more difficult because compared to aesthetic works, "computer programs hover even more closely to the elusive boundary line described in § 102(b)."[86] The Second Circuit characterized the Supreme Court's decision in *Baker v. Selden* as the "doctrinal starting point" for analyzing the scope of copyright in "utilitarian works" because that decision recognized that the copyright in a work describing or illustrating a useful art did not extend to that art.[87] The Second Circuit quoted at length from *Baker* and its holding that Selden's bookkeeping system and the forms that illustrated its use were not protectable by copyright law.[88] The court perceived computer programs to be "roughly analogous" to Selden's book, and consistent with *Baker*, it stated "those elements of a computer program that are necessarily incidental to its function are similarly unprotectable."[89]

To ensure that unprotectable elements of programs would not be inadvertently swept into the infringement determination, the Second Circuit thought it important to fashion a new test for software copyright infringement that would filter those elements out. It announced a three-step test to achieve this objective in cases involving nonliteral software copyright infringement.[90] The first step required constructing a hierarchy of abstractions for the program alleged to be infringed. The second step required filtering out the unprotectable elements of programs. The court said filtration should exclude: aspects of programs that are dictated by efficiency, design choices that are constrained by external factors, and elements of programs that are in the public domain, such as commonplace programming techniques, ideas, and know-how.[91] The *Altai* test's third step called for comparing the "golden nugget[s]" of expression remaining after filtration to determine if there was substantial similarity in the nonliteral expression that the defendant had copied from the plaintiff's program.[92]

Applying this test, the court in *Altai* was satisfied that some similarities between CA's and Altai's programs were due to public domain elements,

85. *Id*. at 705–06.
86. *Id*. at 704.
87. *Id*.
88. *Id*. at 704–05 (quoting Baker v. Selden, 101 U.S. 99, 103 (1879)).
89. *Id*.
90. *See id.* at 706–11.
91. *Id*. at 707–10.
92. *Id.* at 710.

some were at too high a level of generality, and others were "dictated by the functional demands of the program."[93] The parameter list similarities, in particular, were seen as necessary to develop a program that would be compatible with the IBM systems.[94]

The Second Circuit recognized that applying the AFC test to software might mean that copyright would "serve[] as a relatively weak barrier against public access to the theoretical interstices" of program design, but that "results from the hybrid nature of a computer program, which, while it is literary expression, is also a highly functional, utilitarian component in the larger process of computing."[95] The AFC test "not only comports with, but advances the constitutional policies underlying the Copyright Act," the court wrote.[96] CA's economic arguments in favor of broad copyright protection for program SSO were deemed inconsistent with Supreme Court precedents.[97] Adopting CA's theory would, the court added, "have a corrosive effect on certain fundamental tenets of copyright doctrine."[98] Copyright law, the Second Circuit recognized, should not be construed to give utility patent-like protection to highly functional program SSO.[99]

C. A CLOSER LOOK AT *ALTAI*'S FILTRATION FACTORS

The *Altai* decision identified three categories of nonliteral elements of computer programs that should be filtered out of consideration during infringement analysis. The first was "elements dictated by efficiency."[100] Given that Altai did not raise an efficiency defense, the Second Circuit did not need to discuss the exclusion of efficient design elements from the scope of copyright. However, the court was influenced by the "successive filtering" test for software copyright infringement proposed in the Nimmer treatise, which identified efficient program designs as unprotectable elements.[101] The Second Circuit relied upon the merger doctrine to justify

93. *Id*. at 714–15 (quoting Judge Pratt's District Court opinion).

94. *Id*.

95. *Id.* at 712.

96. *Id*. at 711.

97. *Id*. at 711–12 (citing Feist Publ'ns., Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 349–50 (1991)) (rejecting "sweat of the brow" copyright claim in white pages listings of a telephone directory).

98. *Id.* at 712.

99. *Id*.; *see also infra* Part V (discussing why copyrights should not be interpreted to give patent-like protection to program innovations).

100. *Altai,* 982 F.2d at 707–09. Efficiency as a constraint is discussed further *infra* text accompanying notes 344–347.

101. *See Altai*, 982 F.2d. at 707 (citing NIMMER ON COPYRIGHT, *supra* note 10, at 13.03 [F]). The heart of the Nimmer successive filtering test became known as the AFC test after

the filtration of efficient designs insofar as those nonliteral elements of a program were "dictated by considerations of efficiency, so as to be necessarily incidental to that idea."[102] By invoking the merger doctrine to exclude efficient elements of programs, the Second Circuit remained faithful to *Baker v. Selden* and the Supreme Court's holding that functional design elements in copyrighted works are not within the scope of copyright protection, although they may be eligible for patenting.

Altai's exclusion of efficient design elements was an important pronouncement because cases such as *Whelan* had not considered efficiency as a limiting principle in software cases. Under the Third Circuit's conception of software copyright protection in *Whelan*, the question was whether there was *any* other way to carry out a function, and if there was, then the second comer had to do it another way and not copy the plaintiff's SSO.[103] The Second Circuit, however, noted that "[i]n the context of computer program design, the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation."[104] There might well be, ex ante, myriad ways to carry out some functions in a program, but "efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options."[105] The court was reluctant to force programmers to use inefficient (and hence inferior) program designs when efficient ones were available.[106]

---

the *Altai* decision. The court also cited to Professor Menell, *supra* note 10, at 1052, when discussing efficiency as a limiting principle on copyright.

102. *Altai*, 982 F.2d at 707. The "necessary incidents" language, of course, traces back to *Baker*. *See* 101 U.S. 99, 103 (1879).

103. Whelan Associates Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986). The Third Circuit noted that efficiency is a "prime concern" of programmers and the more efficient a program was, the more valuable it would be. *Id*. at 1230. But the court did not perceive efficiency as a design constraint. *See also* Lotus Dev. Corp. v. Paperback Int'l, Inc., 740 F. Supp. 37, 57 (D. Mass. 1990) (rejecting arguments that the functionality of an interface design should limit the scope of copyright protection).

104. *Altai*, 982 F.2d at 708. Computer science books teach programmers about the relative efficiencies of different algorithms and other program design elements for particular types of computations. *See generally* DONALD KNUTH, THE ART OF COMPUTER PROGRAMMING: FUNDAMENTAL ALGORITHMS (3d ed. 1997).

105. *Altai,* 982 F.2d at 708.

106. *See id*. The efficiency exclusion, endorsed in *Altai*, has been influential in subsequent cases. *See, e.g*., Bay State Tech., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1088 (D. Mass. 1996); *see also* Zalewski v. Cicero Builder Dev. Inc., 754 F.3d 95, 105 (2d Cir. 2014) (efficiency considerations limit scope of copyright in architectural works).

The second filtration category concerned those "elements dictated by external factors."[107] Relying heavily on the Nimmer treatise, *Altai* identified five types of "extrinsic considerations" that may "circumscribe[]" a programmer's "freedom of design choice."[108] They were:

> (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is intended to run; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.[109]

Although "dictated by" is the language of merger, as *Altai* recognized in its discussion of efficiency, the Second Circuit pointed to the scenes a faire doctrine as its main doctrinal justification for the "external factors" category.[110] This doctrine excludes from the scope of copyright protection elements that are inevitable or commonly present in works of that kind.[111] Scenes a faire had, the court noted, been employed to limit copyright liability in some prior computer software cases.[112] When applying the AFC test to the facts in *Altai*, the Second Circuit viewed similarities in the organizational charts of the two programs as scenes a faire elements because they "follow[ed] naturally from the work's theme rather than from the author's creativity."[113]

It does seem appropriate to characterize widely accepted programming practices or similarities driven by the demands of the industry as falling within the scenes a faire doctrine.[114] The merger doctrine, however, is more

---

107. *Altai*, 982 F.2d at 709–10.

108. *Id*. at 709.

109. *Id*. at 709–10 (citing NIMMER ON COPYRIGHT, *supra* note 10, at 13-66-71).

110. *Altai*, 982 F.2d at 709–10. The scenes a faire doctrine is distinct from the merger doctrine because the latter focuses constraints on the range of expressive choices available to authors. *See infra* Part IV for an extended discussion of the merger doctrine.

111. *Altai*, 982 F.2d at 709 (citing Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980)) (similarities between book and movie about the Nazi era in Germany were unprotectable scenes a faire elements).

112. *Id*. at 709–10 (citing to four prior cases, including Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204 (9th Cir. 1998)) (similarities in karate videogames were unprotectable as to scenes a faire elements).

113. *Id.* at 715.

114. *See id.* at 710. The Second Circuit pointed to Plains Cotton Coop. Ass'n v. Goodpasture Comput. Serv., Inc., 807 F.2d 1256, 1262 (5th Cir. 1987), which had ruled that similarities in programs were due to demands of the cotton market, to illustrate the point. The Fifth Circuit in *Plains Cotton*, however, did not mention the scenes a faire doctrine in its decision; instead it said that similarities were "dictated by" demands of the industry. *Id.* "Dictated by" is the language of merger.

pertinent to constraints imposed by mechanical specifications of the computer on which a program is run, computer manufacturer design standards, and requirements for achieving compatibility with other programs because these are "necessary incidents" constraints.[115] In applying the AFC test, the Second Circuit quoted the District Court as having found that some similarities between CA's and Altai's programs were "'dictated by the functional demands of the program.'"[116] Dictated by, as noted above, is the language of merger, not of scenes a faire.

The third category of unprotectables consisted of "elements taken from the public domain."[117] This might have seemed unnecessary to say, but perhaps the Second Circuit wanted to make sure that courts would be on the lookout for public domain elements. The *Whelan* decision had been so sweeping in its conception of the scope of copyright in software—every bit of program SSO was said to be protectable expression unless there were no alternative choices possible[118]—that the *Altai* court's reminder that programs, like other works, contain public domain elements was useful. The public domain category of filtration has been used in some post-*Altai* cases,[119] but the Second Circuit did not spell out what kinds of public domain elements it thought should be filtered out.[120]

What is strangely missing from *Altai*'s list of unprotectable elements in computer programs that must be filtered out before deciding whether

115. Some subsequent cases have treated compatibility components of programs as unprotectable under the merger doctrine. *See, e.g.*, Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 540–42 (6th Cir. 2004); *see also* Bateman v. Mnemonics Inc., 79 F.3d 1532, 1544–48 (11th Cir. 1996) (reversing lower court for failure to give proper jury instruction about the possible need to use the same interface code to attain compatibility).

116. *Altai*, 982 F.2d at 714 (quoting Comput. Assocs. Int'l, Inc. v. Altai, Inc.,775 F. Supp. 544, 562 (E.D.N.Y. 1991)).

117. *Id.* at 710.

118. Whelan Associates Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986).

119. *See, e.g.*, Paycom Payroll LLC v. Richison, 758 F.3d 1198, 1205 (10th Cir. 2014) (directing filtration of "ideas, processes, facts, public domain information, merger material, scenes a faire material, and other unprotected elements suggested by the particular facts of the program under examination"); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–43 (10th Cir. 1993) (defendant could lawfully reuse the plaintiff's mathematical constants because they were in the public domain as facts).

120. In support of the public domain category, *Altai* cited Brown Bag Software v. Symantec Software, 960 F.2d 1465, 1473 (9th Cir. 1992) ("Plaintiffs may not claim copyright protection of an expression that is, if not standard, then commonplace in the computer software industry."). This quote would have more suitably illustrated a type 5 external factors excludable under the scenes a faire doctrine in the *Altai* conception of the filtration categories.

infringement has occurred are the categories that § 102(b) renders unprotectable by copyright law: procedures, processes, systems, and methods of operation. The Second Circuit quoted that provision once and cited it in three places.[121] But curiously, it did not consider what kinds of procedures or methods should be unprotectable aspects of programs, let alone direct the filtration of these § 102(b) elements. Later cases have identified algorithms and functional behavior as among the structural elements of programs that must be excluded from protection under § 102(b).[122]

The Second Circuit may have thought the exclusion of procedures, etc., was unnecessary because it regarded the unprotectable elements it did identify as proxies for the procedure, process, system, and method of operation exclusions that § 102(b) says are unprotectable. But merger, scenes a faire, and public domain are different types of limiting principles than processes and methods of operation. If the Second Circuit regarded efficiency, external factors, and public domain elements as proxies for the § 102(b) excludables, it should have explained why the words of the statute should be ignored and why these proxies were appropriate.

It is also possible that the Second Circuit did not perceive a need to filter out these § 102(b) excludables because the *Altai* case arguably did not involve any claim about processes or methods of operation. However, other cases have identified the functional requirements for achieving interoperability with other programs as unprotectable procedures under § 102(b).[123] So this does not explain the omission.

A third possibility—and probably the true explanation—is that the Second Circuit was relying on the Nimmer treatise's successive filtering method for judging software copyright infringement. Because that treatise did not identify procedures, processes, systems, or methods of operation as excludable elements in computer programs, neither did the Second Circuit. The Nimmer treatise has systematically deflected attention away from the wider meaning of § 102(b), treating it as merely a restatement of the idea/expression distinction.[124] Under the influence of the Nimmer treatise,

121. *Altai*, 782 F.2d at 703–04.

122. *See infra* notes 135–42 and accompanying text.

123. *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522, 1526 (9th Cir. 1992).

124. *See* Pamela Samuelson, *Why Copyright Excludes Systems and Processes From the Scope of Its Protection*, 85 TEX. L. REV. 1921, 1953–61 (2007). The Nimmer treatise has ignored the myriad cases that, under the influence of *Baker v. Selden*, have held such things as procedures, processes, systems, and methods of operation are unprotectable elements of copyrighted works. *Id*. at 1936–44 (discussing the cases). Congress intended

courts have sometimes been reluctant to pay attention and give content to these exclusions, even though these elements have a wider range of possible applications for computer programs than for any other type of copyrighted work.[125] Procedures, processes, systems, and methods of operation are, after all, functional design elements of the "essentially utilitarian nature of computer program[s]."[126]

III. CONCEPTUALIZING THE PROPER ROLE OF § 102(b) IN COMPUTER PROGRAM COPYRIGHT CASES

Because computer programs embody many procedures, processes, systems and methods of operation that lie beyond the scope of copyright, it would be logical for § 102(b) to have a significant, and perhaps even a central, role in the analysis of claims of software copyright infringement. One way to accomplish this objective would be to adapt the *Altai* AFC test, as some courts have already done, by adding a fourth category of program design elements derived from § 102(b) that should be filtered out before proceeding with the final stage of infringement analysis.[127] Another way would be, in appropriate cases, to consider and apply the functional design element exclusions in § 102(b) without tying them to the *Altai* framework, which other courts have also already done.[128]

Section A discusses five propositions about § 102(b) that should be uncontroversial and shows that courts in recent years have been more receptive to employing § 102(b) as a limiting principle of software copyright law. Sections B and C review the *Lotus v. Borland* and *Oracle v. Google* cases, and explain why the application of § 102(b) in those cases has generated some controversy. It is unfortunate that the Supreme Court split 4-4 in *Borland* in 1995 and that twenty years later, it declined to hear Google's appeal; the split in the circuits as to the proper interpretation of § 102(b) is unlikely to be resolved any time soon. Section D explains why

---

to codify these common law exclusions from the scope of copyright by including § 102(b) in the statute *Id*. at 1944–52 (discussing the legislative history of § 102(b)); *see also* Ann Bartow, *The Hegemony of the Copyright Treatise*, 73 U. CINN. L. REV. 581 (2004) (criticizing courts for relying heavily on copyright treatises, including Nimmer's, in interpreting copyright law).

125. The *Franklin* and *Whelan* decisions are exemplary of this trend. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1250–52 (3d Cir. 1983); Whelan Associates Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1234–36 (3d Cir. 1986).

126. *Altai*, 982 F.2d at 704.

127. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 836–37 (10th Cir. 1993); *see also infra* text accompanying note 453.

128. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–16 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

the Federal Circuit's interpretation of § 102(b) compatibility defenses in *Oracle* should be repudiated in subsequent software copyright cases.

A. FIVE UNCONTROVERSIAL PROPOSITIONS ABOUT § 102(b)

This Section articulates five propositions about § 102(b) that should be uncontroversial. First, § 102(b) should not be interpreted in a manner that would deprive programs of any copyright protection. Second, the procedure, process, system, and method of operation exclusions in § 102(b) are meaningful limits on the scope of copyright protection available to programs. Third, one function of the § 102(b) exclusion of processes and methods from the scope of copyright is to maintain boundaries between the copyright and patent regimes. Fourth, the utilitarian nature of programs differentiates them from conventional literary works because they contain functional design elements such as processes that are excluded under § 102(b). Fifth, "SSO" is not a useful way to distinguish between those nonliteral elements of programs that are unprotectable under § 102(b) and those that constitute protectable expressions.

1. Section 102(b) Does Not Exclude Program Code from Protection

The least controversial proposition about § 102(b) in relation to computer programs is that Congress could not possibly have intended courts to give a completely literal interpretation to § 102(b) because this would render programs ineligible for copyright protection. Object code is unquestionably a functional process. Therefore, the Third Circuit in *Franklin* correctly rejected Franklin's § 102(b) defense: Franklin copied the Apple OS programs, bit for bit, and did not even try to reimplement the functionality of the Apple programs in independently written code.[129] Furthermore, while object code is the most functional embodiment of program processes, because source code forms of programs function as detailed statements and instructions for carrying out certain tasks, they are unquestionably "procedures" within the normal meaning of that word. Yet, to respect Congress' decision to extend copyright protection to computer programs, original program code, whether in source or object code form, should generally be protectable by copyright law.[130]

---

129. *See supra* note 47 and accompanying text.

130. Occasionally, a program may either have insufficient originality to support a copyright or be rendered unprotectable because function and expression have merged. *See, e.g.*, Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 540–42 (6th Cir. 2004) (questioning the originality of a program embedded in a printer cartridge, but also applying the merger doctrine to it).

2. *The Procedure, Process, System and Method of Operation Exclusions of § 102(b) Must Mean Something*

A second uncontroversial proposition is that the words of exclusion from the scope of copyright protection for "procedure[s], process[es], system[s], [and] method[s] of operation" in § 102(b) must mean something.[131] As a matter of logic, they cannot be synonymous with the "idea" exclusion because "idea" is but one of eight categories of excludable elements listed in § 102(b). One term cannot subsume the other seven. Giving meaning to each of the statutory exclusions is consistent with conventional canons of statutory construction.[132] When a statute specifically identifies several categories of unprotectable elements and says that "[i]n no case" should any of them be within the scope of copyright protection,[133] courts should not read all but one of the terms out of the statute, as the appellate courts did in *Whelan* and *Oracle*.

That these functional design elements excluded by § 102(b) should be given close attention in software copyright infringement cases is also evident from the legislative history of the 1976 Act. Both the House and Senate Reports explained why § 102(b) was put in the statute:

> Some concern has been expressed lest copyright in computer programs should extend protection to the *methodology or processes* adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the *actual processes or methods* embodied in the program are not within the scope of the copyright law.[134]

Courts in software copyright cases have identified several types of nonliteral elements of programs as unprotectable procedures, processes, systems, or methods of operation under § 102(b). Among the elements that

131. 17 U.S.C. § 102(b).

132. *See* Montclair v. Ramsdell, 107 U.S. 147, 152 (1883) ("It is the duty of the court to give effect, if possible, to every clause and word of a statute, avoiding, if it may be, any construction which implies that the legislature was ignorant of the meaning of the language it employed."); LARRY M. EIG, CONG. RESEARCH SERV., 97-589, STATUTORY INTERPRETATION: GENERAL PRINCIPLES AND RECENT TRENDS 13 (2011) ("The modern variant [of this principle] is that statutes should be construed 'so as to avoid rendering superfluous' any statutory language . . . ." (quoting Hibbs v. Winn, 542 U.S. 88, 101 (2004))).

133. 17 U.S.C. § 102(b).

134. H.R. REP. NO. 94-1476, at 57 (1976); S. REP. NO. 94-473, at 54 (1975) (emphasis added).

courts have filtered out are algorithms,[135] mathematical constants,[136] rules editing methods,[137] methods of calculation,[138] command structures,[139] data structures,[140] interfaces necessary to interoperability,[141] and functional program behavior.[142] Also noteworthy is the Copyright Office's articulation of functional elements of computer programs that the Office regards as unprotectable by copyright law.[143]

The § 102(b) exclusions apply, of course, to all types of copyrighted works, not just to computer programs. Procedures, processes, systems and methods of operation have, in fact, been excluded from the scope of copyright protection both before and after § 102(b) was added to the statute. Consider, for example, *Brief English Systems v. Owen* which illustrates the system exclusion.[144] The Second Circuit decided that Owen was free to write his own book on the shorthand system that the plaintiff had devised because that system was ineligible for copyright protection.[145] *Taylor Instrument Cos. v. Fawley-Brost Co*. illustrates the method of operation

135. *See, e.g.*, Torah Soft Ltd. v. Drosnin, 136 F. Supp. 2d 276, 291 (S.D.N.Y. 2001).

136. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–43 (10th Cir. 1993). *Gates Rubber* directs use of a "process/expression" distinction in computer program cases. *Id*.

137. *See, e.g.*, Ilog v. Bell Logic, LLC, 181 F. Supp. 2d 3, 14 (D. Mass. 2002).

138. *See, e.g.*, Harbor Software v. Applied Sys., Inc., 925 F. Supp. 1042, 1052 (S.D.N.Y. 1996).

139. *See, e.g.*, Mitek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1557 (11th Cir. 1996).

140. *See, e.g.*, Baystate Techs., Inc. v. Bentley Systems, Inc., 946 F. Supp. 1079, 1088–89 (D. Mass. 1996).

141. *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992).

142. *See, e.g.*, O.P. Solutions v. Intell. Prop. Network Ltd., No. 96 Civ. 7952 (LAP), 1999 WL 47191, at *16–20 (S.D.N.Y. Feb. 2, 1999).

143. Compendium of U.S. Copyright Practices (3d ed. 2015) [hereinafter Compendium], § 721.7 ("[T]he Office will not register the functional aspects of a computer program, such as the program's algorithm, formatting, functions, logics, system design, and the like."); *see also id*. at § 721.9(J) (indicating that registration will not be accepted if the program author claims copyright in an algorithm, computation, data, formatting, formulas, interfaces, language, layout, logic, menu screens, models, organization, protocols, and system design, among other categories of unprotectable elements).

144. 48 F.2d 555 (2d Cir. 1931).

145. *Id*. at 556 ("There is no literary merit in a mere system of condensing written words into less than the number of letters usually used to spell them out. Copyrightable material is found, if at all, in the explanation of how to do it."). Numbering systems for hardware parts are similarly unprotectable by copyright law. *See, e.g.*, ATC Distrib. Group, Inc. v. Whatever It Takes Transmission & Parts, Inc., 402 F.3d 700 (6th Cir. 2005); Southco, Inc. v. Kanebridge Corp., 390 F.3d 276 (3d Cir. 2004).

exclusion.[146] The Seventh Circuit rejected Taylor's claim of copyright in charts designed for use as part of a method of operating a temperature recording machine.[147] *Publications Int'l Ltd. v. Meredith Corp*. illustrates the exclusion of procedures from the scope of copyright.[148] The Seventh Circuit held that individual recipes were unprotectable procedures under § 102(b), and others could freely copy them.[149] *Bikram's Yoga College of India, L.P. v. Evolation Yoga, LLC,* illustrates the process exclusion.[150] The Ninth Circuit held that the series of yoga poses and breathing exercises that Bikram Choudhury had developed were unprotectable processes under § 102(b) because of their functional character in helping to attaining psychological and spiritual well-being.[151]

3. The Process and System Exclusions of § 102(b) Are Partly Aimed at Maintaining Boundaries between Copyright and Patent Laws

A third proposition about § 102(b) that should be uncontroversial is that procedures, processes, systems, and methods of operation are excluded from the scope of copyright protection in part to maintain a balance between the role of copyright in protecting authorial expression and the role of patent law in protecting inventions in the useful arts.[152] Patented processes, systems and methods of operation are often described in documents or

146. 139 F.2d 98 (7th Cir. 1943); *see also* Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947) (upholding the Copyright Office's refusal to register charts for recording data).

147. *Taylor Instrument,* 139 F.2d at 100–01; *see also* Coates-Freeman Assocs., Inc. v. Polaroid Corp.792 F. Supp. 879, 884–85 (D. Mass. 1992) (problem-solving method unprotectable by copyright law).

148. 88 F.3d 473 (7th Cir. 1996).

149. *Id*. at 480–81.

150. 803 F.3d 1032 (9th Cir. 2015); *see also* Palmer v. Braun, 287 F.3d 1325, 1333–34 (11th Cir. 2002) (citing to § 102(b) and affirming denial of a preliminary injunction because similarities between Palmer's and Braun's courses and course materials were largely due to the fact that they were teaching the same processes for raising human consciousness).

151. *Bikram's Yoga*, 803 F.3d at 1039–40.

152. *See, e.g*., Incredible Techs., Inc. v. Virtual Techs., Inc., 284 F. Supp. 2d 1069, 1078 (N.D. Ill. 2003), *aff'd*, 400 F.3d 1007 (7th Cir. 2015) (control panel features of videogame were "potentially patentable-but not copyrightable"); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1541, 1546 (11th Cir. 1996); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 837 (10th Cir. 1993) (recognizing that some program processes may be patentable). Processes are one of four categories of statutory subject matters eligible for patent. *See* 35 U.S.C. § 101. Claims for patentable processes are often stated as methods for carrying out certain operations. Patents for machines are typically stated as systems to perform certain functions.

illustrated in drawings. But "the principle is the same in all," as the Supreme Court said in *Baker v. Selden* more than a hundred twenty years ago:[153] "The description of the [useful] art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself . . . . [That] can only be secured, if it can be secured at all, by letters-patent."[154] It would be "a surprise and fraud on the public," the Supreme Court wrote, if Selden could get through the copyright in his book a longer duration of exclusive rights than if he had been able to get the patent he sought (which he apparently failed to obtain).[155]

In *Taylor Instrument*, the Seventh Circuit invoked *Baker* in deciding that Taylor's charts for recording temperatures over time were not proper subject matter for copyright protection.[156] Although Taylor had obtained registration certificates from the Copyright Office, the charts were unprotectable by copyright law because they were indispensable parts of Taylor's recording machines.[157] Fawley-Brost, the alleged infringer, was entitled to provide a competitive alternative to those customers who owned Taylor's temperature recording machines and wanted cheaper charts that would interoperate with the Taylor machines. The Seventh Circuit observed:

> While it may be difficult to determine in which field [of intellectual property] protection must be sought, it is plain, so we think, that it must be in one [copyright] or the other [patent]; it cannot be found in both. In other words, there is no overlapping territory, even though the line of separation may in some instances be difficult of exact ascertainment.[158]

153. Baker v. Selden, 101 U.S. 99 (1879).

154. *Id.* at 105; *see also Bikram's Yoga*, 803 F.3d at 1039–40 (ruling yoga healing methodology is uncopyrightable, but stating that if it is "entitled to protection at all, that protection is more properly sought through the patent process").

155. *Id.*; *see* Pamela Samuelson, *Baker v. Selden: Sharpening the Distinction Between Authorship and Invention*, *in* INTELLECTUAL PROPERTY STORIES 160–61 (Rochelle C. Dreyfuss & Jane C. Ginsburg eds., 2005) (noting that the preface to Selden's book referred to his patent application). Had the Court accepted Selden's claim of copyright protection in the bookkeeping system, he could have had exclusive rights in it for up to forty-two years instead of the much shorter duration available had a patent issued for the system.

156. 139 F.2d 98, 99–100 (7th Cir. 1943).

157. *Id.* at 100.

158. *Id.* at 99; *see also* Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947) (only patent, not copyright, protection was available to charts as parts of recording machines). Note that *Taylor Instrument* and *Brown Instrument* are pre-software interoperability cases.

Similar considerations underlay the Ninth Circuit decision to reject Sega's infringement claim in *Sega Enters. Ltd. v. Accolade, Inc*.[159] The infringement suit commenced after Accolade reverse engineered Sega programs to get access to information about the functional requirements for achieving interoperability with Sega's Genesis platform. Sega sued to stop Accolade, arguing that the copies made in the course of reverse engineering were infringements.[160] The Ninth Circuit stated that "[i]f disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection by Congress" under § 102(b).[161] The court characterized the Sega interface as an unprotectable procedure under § 102(b), saying that to get an exclusive right in that interface, Sega needed to get a patent.[162] This ruling allowed Accolade (and other independent software developers) to create new non-infringing programs that could run on the popular Genesis platform and maintained the proper boundary lines between patent and copyright protections for computer programs.

4. Because of § 102(b) Exclusions, the Scope of Copyright in Programs Is Thinner than the Scope of Copyright in Conventional Literary Works

A fourth proposition concerning § 102(b) that should be uncontroversial is that the scope of copyright protection in computer programs is generally much thinner than the scope of copyright in conventional literary works (e.g., novels and poetry) because programs embody many functional design elements that lie outside the scope of copyright protection under § 102(b).[163] Courts have recognized that it is necessary to filter out procedures, processes, systems, and methods of operation before ruling on software copyright infringement claims.[164] Conventional literary works, by contrast, are typically highly expressive and non-functional, which is why courts

159. *See* 977 F.2d 1510, 1524–27 (9th Cir. 1992) (noting that the functionality of programs limits scope of copyright under § 102(b) and emphasizing that copyright should not be construed to give programmers patent-like protection for elements excluded from copyright under § 102(b)).

160. *Id.* at 1516–17. Sega did not claim that Accolade infringed because it copied the SSO of the Genesis interface.

161. *Id*. at 1526.

162. *Id*. at 1522, 1526.

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

164. *See, e.g*., Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–46 (10th Cir. 1993) (applying the abstraction-filtration-comparison test to the computer program).

mainly concentrate on filtration of ideas, facts, and scenes a faire elements in assessing infringement claims.[165]

5. *SSO Obscures the Distinction between Nonliteral Elements of Programs That Are Protectable by Copyright and Those That Are Unprotectable Under § 102(b)*

A fifth proposition that should be uncontroversial is that "SSO" is not a useful way of conceptualizing which nonliteral elements of computer programs are copyright-protectable expressions that cannot be reused without infringing.[166] Although courts have sometimes based infringement findings on copying of the detailed structure of novels or plays,[167] SSO was not a term of art in the copyright field prior to the *Whelan* decision. Nor is SSO a term of art in the computing field, as the Second Circuit observed in *Altai*.[168] As applied to programs, SSO obscures the reality that many aspects of program structure and organization are procedures, processes, systems, and methods of operation that § 102(b) excludes from copyright's protection.[169] The term fails to provide a workable framework within which courts can separate out nonliteral program expression from nonliteral functional elements excluded under § 102(b). As the District Court in *Oracle* observed, there has been a trend away from use of SSO in the post-*Altai* software copyright case law, which has been "driven by fidelity to Section 102(b)" to avoid the "danger" of overprotection of programs by copyright law.[170]

---

165. This explains why Judge Hand's patterns-of-abstraction test for infringement, which has long been applied to conventional literary and dramatic works, does not mention processes or systems. *See* Nichols v. Universal Pictures, 45 F.2d 119, 121 (2d Cir. 1930), *cert. denied*, 282 U.S. 902 (1931).

166. *See Altai*, 982 F.2d at 702–06 (questioning the *Whelan* decision's use of SSO); Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 996 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014) (pointing out that SSO terminology had not been used in the software copyright case law since 1989); *see also* Weinreb, *supra* note 10, at 1170 (critical of SSO formulation for software).

167. *See, e.g.*, Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936).

168. *Altai*, 982 F.2d at 706 (noting that SSO reflects an inaccurate understanding of computer science).

169. Plaintiffs in software copyright cases sometimes rely on the literary work metaphor to deflect attention from the functionality of software. *See, e.g.*, Opening Brief and Addendum of Plaintiff Appellant, Oracle of Am. v. Google Inc., Court of Appeals for the Federal Circuit, Case No. 13-1021, at 1–2 (likening the Android software to a knockoff of a Harry Potter novel).

170. *Oracle*, 872 F. Supp. 2d at 996.

6. Summary

These five propositions are grounded in a straightforward reading of the copyright statute and long-standing policy considerations that explain why copyright protection is "thinner" for utilitarian works than for conventional works of art and literature. Ongoing progress in the computing field requires a realm of freedom to reuse functional design elements of programs.[171] Courts and commentators who believe that § 102(b) merely restates the idea/expression distinction should reconsider their conceptions of § 102(b).

But even accepting these propositions, the question remains whether the command structures in the *Borland* and *Oracle* cases should have been deemed unprotectable systems or methods of operation under § 102(b). To that question, we now turn.

B. *LOTUS V. BORLAND*

Lotus v. Borland is the best known of the cases addressing whether a command structure of a computer program user interface (UI) is protectable by copyright law.[172] Lotus was not, however, the first software developer to sue a competitor for copyright infringement because the defendant adopted the same set of commands, organized in the same way, as the plaintiffs' program in order to carry out the same set of program functions.[173]

Courts in the early cases typically found infringement because they conceived of the plaintiffs' command structures as compilations of words whose selection and arrangement was original enough to warrant copyright protection, even if each command word on its own (e.g., copy or paste) was unprotectable.[174] These cases often relied upon *Whelan*, which considered

171. *See, e.g.*, Altai, 982 F.2d at 707–12 (emphasizing various elements of programs that are unprotectable by copyright law because of the need of programmers to reuse them and quoting commentary expressing concern that *Whelan* would enable programmers to lock-up basic programming techniques and give first comers quasi-monopoly power).

172. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995).

173. After *Whelan*, several lawsuits were brought against makers of "clone" or "work-alike" programs that used the same or substantially similar commands as the industry leader. *See, e.g.*, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990); Digital Commc'n Assoc., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449 (N.D. Ga. 1987); Broderbund Software, Inc. v. Unison World, 648 F. Supp. 1127 (N.D. Cal. 1986). These were sometimes known as "look and feel" cases because of similarities in the way the programs looked and the way they operated in the same or substantially similar ways. *See, e.g.*, Pamela Samuelson, *Why the Look and Feel of Software User Interfaces Should Not Be Protected by Copyright Law*, 32 COMM. ACM 563 (May 1989). *Whelan* expressed receptivity to copyright protection for program "look and feel." Whelan Assoc. Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1231, 1245–47 (3d Cir. 1986).

174. *See, e.g.*, *Softklone*, 659 F. Supp. at 452–59.

all structural elements of programs (e.g., UI command hierarchies) as protectable expression as long as there was more than one way to structure those elements.[175]

Borland unquestionably copied the command hierarchy of Lotus 1-2-3 in its Quattro Pro (QP) program.[176] The District Court regarded Borland's appropriation of this hierarchy as infringement.[177] Even when Borland issued a new release of the QP emulation UI so that the Lotus command words were no longer visible, the District Court found this too infringed because QP was still utilizing the Lotus command structure, albeit invisibly.[178] The District Court conceived the command hierarchy to have, in effect, become a nonliteral element of the program.

Borland's reuse of the command structure of Lotus 1-2-3 was different from the earlier UI cases. Unlike those defendants, Borland had developed its own "native" UI for QP, which organized commands in a different way than Lotus 1-2-3. But Borland provided an "emulation" UI that presented users with the same commands in the same order as 1-2-3 so that prospective customers who had built macros (i.e., mini-programs for commonly executed sequences of spreadsheet functions) in the Lotus macro language could use those macros in QP.

Because the command hierarchy of Lotus 1-2-3 was "a fundamental part of the functionality of the Lotus macros,"[179] Borland argued it was outside the scope of copyright protection available to the Lotus program under § 102(b).[180] This structure, Borland believed, was akin to the structure of Selden's bookkeeping forms, and as constituent elements of Lotus' method of operation or system, it was patent, not copyright, subject matter.[181]

---

175. *See, e.g.*, *id*. at 454–55.

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

177. Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 219 (D. Mass. 1992), *rev'd*, 49 F.3d 807 (1st Cir. 1995).

178. *See generally* Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 223, 245 (D. Mass. 1993).

179. In *Lotus Dev. Corp. v. Paperback Software Int'l*, the District Court recognized the functionality of the Lotus macro system, which allowed users to construct mini-programs for commonly executed sequences of functions, but did not think it was relevant to copyright protection for the command hierarchy. 740 F. Supp. 37, 64 (D. Mass. 1990). Paperback had argued that macro compatibility required use of this hierarchy, but the court found this unpersuasive. *Id*. at 69.

180. Borland argued that *Altai* supported its macro-compatibility defense. *Borland*, 799 F. Supp. at 215–19.

181. Borland's argument is most clearly set forth in the First Circuit's opinion. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 814 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

The District Court did not find the user macro interoperability argument persuasive.[182] It viewed Borland's copying of the 1-2-3 command structure as infringement because the command hierarchy was not an abstract idea, but rather concrete and detailed; in that court's view, there were other ways that spreadsheet commands could be named and organized, as evidenced by QP's native UI.[183]

Borland appealed its loss to the First Circuit.[184] The main question on appeal was what meaning to give to the exclusions set forth in § 102(b).[185] Given the text of the statute and the legislative history explaining Congress's reasons for including § 102(b) in the statute, it seemed unlikely that this court would rule that § 102(b) was merely a restatement of the idea/expression distinction, as the District Court had opined in *Borland*.

In its ruling in Borland's favor, the First Circuit held that the command structure was a method of operation under § 102(b), although on a different rationale than Borland argued.[186] The First Circuit likened the command hierarchy of Lotus 1-2-3 to the buttons for interacting with a VCR machine.[187] Both were, in its view, methods of operating machines to accomplish functional tasks.[188] As methods of operation, both were unprotectable by copyright law. Although the First Circuit invoked *Baker* in support of its holding, there was disappointingly little analysis in support of the court's conclusion, either about the implications of *Baker* or the meaning that courts should give to § 102(b).[189]

The First Circuit viewed Borland as having literally copied the Lotus command structure, but decided that didn't matter because that structure, however creative it might be, constituted a method of operation.[190] It did

182. *Borland*, 799 F. Supp. at 210 ("[I]t is irrelevant that the 1-2-3 interface includes functional elements or 'comprises a system' so long as it includes separable expressive elements."). In its earlier ruling against Paperback, which had made a clone of Lotus 1-2-3, the District Court dismissed as a "word game" Paperback's argument that the Lotus macro language was unprotectable by copyright law. *Paperback*, 740 F. Supp. at 72.

183. *Borland*, 799 F. Supp. at 212–13.

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

185. *Id.*

186. Borland argued that the facts and arguments in *Baker* were "identical" to those it was making. The First Circuit was not convinced. *Id.* at 814.

187. *Id.* at 817.

188. For a similar metaphor, see Apple Computer v. Microsoft Corp., 799 F. Supp. 1006, 1023 (N.D. Cal. 1992), *aff'd*, 35 F.3d 1435 (9th Cir. 1994) (likening Apple's graphical user interface to the user interface for automobiles).

189. *Borland*, 49 F.3d at 813–14; *see also* Weinreb, *supra* note 10, at 1207 (describing the First Circuit's analysis as "too short to be satisfactory").

190. *Borland*, 49 F.3d at 815–16.

not, however, find persuasive Borland's *Baker*-based argument that the command structure was patent, not copyright, subject matter.[191] The First Circuit cited approvingly to *Altai*, but regarded the Second Circuit's decision as inapplicable because that case involved claims of non-literal infringement, whereas *Borland* was, in its view, a literal copying case.[192] Although literal copying of content from a copyrighted work is generally more likely to be infringement, the First Circuit in *Borland* conceptualized the command hierarchy as too functional to qualify as protectable expression.[193] *Altai*, in fact, had more to teach the First Circuit on compatibility issues than the court perceived in *Borland*.[194]

Whether the 1-2-3 command hierarchy was a literal or nonliteral element of the Lotus program is actually a more interesting question than either the District Court or the First Circuit recognized.[195] In some sense it was both: users of the Lotus program can, of course, see the selection and arrangement of command words when they use 1-2-3. Viewed as a compilation of words, the command hierarchy looks like a literal component of the program. Each command is, however, an abstraction that not only identifies the particular function that the command represents, but also provides users with a means to invoke that function, which, in turn, is a nonliteral element embedded in the literal text of the program. U.S. copyright law defines "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."[196] Viewed in this light, the program source and object code is the literal expression of the software, and the UI command hierarchy is an abstraction that identifies the set of functions that the

191. *Id*. at 813–14. The First Circuit did not explain why it found the argument unpersuasive. The patent or copyright subject matter issue is discussed *infra* Part V.

192. *Id*. at 814–15.

193. *Id*. at 815 ("The Lotus command hierarchy provides the means by which users control and operate Lotus 1-2-3.").

194. *See* Brief Amici Curiae of American Committee for Interoperable Systems and Computer & Communications Industry Ass'n in Support of Respondent, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728487 (explaining the relevance of *Altai* in the *Borland* case). Later cases, relying in part on *Altai*, recognized that even literal copying may be excusable when necessary for achieving interoperability. *See, e.g.*, Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1547 (11th Cir. 1996).

195. Weinreb noted the oddity of the parties' positions about the nature of programs that copyright could protect: Lotus taking a broad conception of programs, wanting to tie its UI closely to the code, and Borland taking a narrow view, as though the code constituted the program and the UI was a result of the program. Weinreb, *supra* note 10, at 1154–63.

196. 17 U.S.C. § 101 (definition of "computer program").

program is capable of executing. The UI command structure is, moreover, among the results that the program code produces.

The nonliteral character of the command hierarchy should have been evident when Borland introduced the key reader version of QP, which allowed macros constructed in 1-2-3 to be executed, even though users could no longer see the commands.[197] The First Circuit should have recognized the macro system as a nonliteral element of the Lotus program because it was neither visible to a 1-2-3 user through the UI, nor was it associated with particular blocks of code.[198]

When the Supreme Court decided to hear Lotus's appeal, it seemed that the Court would finally provide an answer to the long-simmering question of how courts should interpret the method and system exclusions of § 102(b) as applied to computer programs. However, the Court's 4-4 split affirmed the First Circuit ruling without setting a precedent or resolving the circuit split on this issue.[199]

The deep split within the Court may have been due to some Justices agreeing with the literal infringement approach taken by the District Court, while other Justices may have seen some merit in the First Circuit's interpretation of § 102(b), as cryptic as that court's analysis was, or in Borland's argument, that the command hierarchy was patent, not copyright, subject matter.[200]

The sounder analysis supporting the ruling in Borland's favor would have focused on the essential role that the Lotus command structure played in facilitating the functionality of the Lotus macro system. As the District Court recognized, the command hierarchy was a "fundamental part of the

197. Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 223, 228 (D. Mass. 1993) ("[T]he Key Reader file contains a virtually identical copy of the Lotus menu tree structure, but represented in a different form and with first letters of menu command names in place of the full menu command names."). The District Court found infringement of the key reader version of QP. *Id.* at 231. This, in effect, extended copyright protection to the functional behavior of the program. *See* Pamela Samuelson, *Brief Amicus Curiae of Copyright Law Professors* in Lotus Development Corp. v. Borland Int'l, Inc. (brief to U.S. Supreme Court), 3 J. INTELL. PROP. L. 103, 131–32 (1996) (explaining why the District Court's ruling would have extended protection to the functional behavior of programs and why copyright should not extend so far).

198. An analogy may help to clarify the relationship among these components: The UI of a program is akin to the face of a clock. The program code is like the mechanism inside the clock that causes the hands of the clock to move. The macro system is like a specialized part of the clock (e.g., the component that users can set to cause an alarm to ring at a particular time).

199. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1995).

200. Brief for Respondent at 22–37, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728538.

functionality of the Lotus macros."[201] That is, the Lotus command structure was a critical component in the functioning of the macros because user-created macros would not execute in QP unless exactly the same commands were available, arranged in exactly the same order.[202] The command structure was thus a constituent element of the macros system that should have been outside the scope of the Lotus copyright under § 102(b).[203] Alternatively, the Court could have recognized that, in keeping with *Altai*, there were external factors (i.e., the investment of users who had constructed macros in the Lotus macro language) that constrained the design choices of second comers (i.e., Borland because the command structure was essential to achieving compatibility with user macros).[204]

Judge Boudin's concurring opinion in *Borland* suggests the macro compatibility consideration resonated with him:

> [I]t is very hard to see that Borland has shown any interest in the Lotus menu except as a fall-back option for those users already committed to it by prior experience or in order to run their own macros using 1-2-3 commands . . . [I]t is unlikely that users who value the Lotus menu for its own sake—independent of any investment they have made themselves in learning Lotus'

201. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 64 (D. Mass. 1990).

202. *See* Pamela Samuelson, *Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of* Lotus v. Paperback, 55 LAW & CONTEMP. PROB. 311, 331–37 (1992) (explaining why the Lotus macro system was unprotectable under § 102(b)); *see also* Brief Amicus Curiae of Copyright Professors in Support of Respondent at 3–5, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728563 (setting forth alternative theories about the application of § 102(b) to the Lotus command hierarchy).

203. That methods and systems and their constituent parts are unprotectable by copyright law was recently affirmed in the Ninth Circuit's *Bikram's Yoga* decision. *See Bikram's Yoga*, 803 F.3d at 1039 ("An essential element of this 'system' is the order in which the yoga poses and breathing exercises are arranged."); *id.* at 1042 ("[T]he medical and functional considerations at the heart of the [Bikram] Sequence compel the very selection and arrangement of poses and breathing exercises for which he claims copyright protection.").

204. *See* Brief Amici Curiae of American Committee for Interoperable Systems and Computer & Communications Industry Ass'n in Support of Respondent, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728487 (explaining the relevance of *Altai* in the *Borland* case). A second alternative analysis would have framed *Borland* as an instance in which functionality and expression had, in effect, merged because of the role of the Lotus command hierarchy in enabling the functionality of the macro system. *See, e.g.*, Brief of the United States as Amicus Curiae at 19–21, Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2015 WL 2457656 (suggesting a merger rationale for the *Borland* ruling); see also *infra* text accompanying note 382 for a discussion of merger in relation to *Borland*.

> commands or creating macros dependent upon them—would choose the Borland program in order to secure access to the Lotus menu.[205]

Judge Boudin also took note of the lock-in effects that would result from a ruling in Lotus' favor:

> [I]t is hard to see why customers who have learned the Lotus menu and devised macros for it should remain captives of Lotus because of an investment in learning made by the users and not by Lotus. Lotus has already reaped a substantial reward for being first; assuming that the Borland program is now better, good reasons exist for freeing it to attract old Lotus customers: to enable the old customers to take advantage of a new advance, and to reward Borland in turn for making a better product. If Borland has not made a better product, then customers will remain with Lotus anyway.[206]

It is unfortunate that Judge Boudin did not find a way to connect this concern to the Congressional intent that § 102(b) serve as a statutory tool through which courts could take competition and ongoing innovation policy considerations into account in construing the scope of copyright protection in programs.[207] He considered the majority's interpretation to be "defensible," although he thought that fair use would be a closer doctrinal fit, suggesting there was need for a new doctrine to address such considerations.[208]

A compatibility-based § 102(b) argument should have prevailed in *Borland*. Subsequent cases have done a somewhat better job than the First Circuit in employing § 102(b) in software copyright cases.[209] The Ninth Circuit reaffirmed in 2000 that program interfaces necessary for achieving interoperability are unprotectable procedures under § 102(b),[210] and has

---

205. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996) (Boudin, J., concurring).

206. *Id*. at 821; *see also* Dan L. Burk, *Method and Madness in Copyright Law*, 2007 UTAH L. REV. 587, 591–92 (2007).

207. *See supra* note 4 and accompanying text.

208. *Borland*, 49 F.3d at 821–22 (Boudin, J., concurring).

209. *See, e.g.*, Incredible Techs. v. Virtual Techs., Inc., 284 F. Supp. 2d 1069, 1078 (N.D. Ill. 2003), *aff'd*, 400 F.3d 1007, 1012 (7th Cir. 2005); Mitek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1556–57 (11th Cir. 1996); Ilog, Inc. v. Bell Logic LLC, 181 F. Supp. 2d 3, 12–14 (D. Mass. 2002); Torah Software Ltd. v. Drosnin, 136 F. Supp. 2d 276, 291–92 (S.D.N.Y. 2001); O.P. Solutions v. Intell. Prop. Network Ltd., No. 96 Civ. 7952 (LAP), 1999 WL 47191, at *16–18 (S.D.N.Y. Feb. 2, 1999).

210. Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602–03 (9th Cir. 2000).

also held that a list of commands for the user interface of a computer program is unprotectable by copyright law.[211]

It is fair to say that the First Circuit could have done a better job explaining why the Lotus command hierarchy was an unprotectable system or method of operation under § 102(b). This Section has explained that this ruling was sound and consistent with the competition and innovation policies expressed in other software copyright decisions that have construed § 102(b) as a limiting principle on the scope of copyright when compatibility issues are at stake.

C. *ORACLE V. GOOGLE*

Google's confidence that it could lawfully use command structures derived from the Java API was built partly on the *Borland* decision's interpretation of § 102(b) and partly on the Ninth Circuit's ruling that the Sega interface was unprotectable under § 102(b).[212] In the twenty years since the Supreme Court punted on interpreting § 102(b) in *Borland*, the First Circuit's decision has been met with mostly positive reactions in the

211. Ashton-Tate Corp. v. Ross, 916 F.2d 516, 521–22 (9th Cir. 1990) (rejecting Ross's claim of joint authorship based on having provided a list of commands for a computer program on which he and another programmer were working, and agreeing with the District Court that the list of commands was unprotectable under § 102(b)).

212. *See* Brief of Appellee and Cross-Appellant Google Inc., Oracle Am., Inc. v. Google Inc., No. 2013-1021 at 33–52 (relying on *Sega v. Accolade*), 57–65 (relying on *Lotus v. Borland*) (Fed. Cir. 2013). Oracle sued Google in 2010 for infringing patents that Oracle claimed read onto component elements of the Android mobile phone platform and for infringing Oracle's copyright in the Java platform, which was said to include code, specifications, documentation and other materials. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1347–48 (Fed. Circ. 2014). The jury ruled against Oracle's patent claims and against its claims of copyright infringement in Java documentation; however, the jury found Google to have infringed copyright in a range by checking the subroutine and in structure of certain Java API packages (assuming that these were copyright-protectable, an issue which the District Court reserved for itself). *Id.* Oracle's appeal of the copyright ruling went to the CAFC because of the patent claim in the complaint. The CAFC acknowledged that it was obliged to follow Ninth Circuit precedent. *Id.* at 1353. This section will show that it did not do so.

courts.[213] Many have followed it,[214] some have distinguished it,[215] and a few have criticized or rejected it.[216] The trial court in *Oracle* was one of the followers.[217] The CAFC, on the other hand, both distinguished and criticized *Borland* in its decision in favor of Oracle.[218]

Although the facts of the *Oracle* case are more complicated than those in *Borland*, the legal question in the two cases is remarkably similar: whether a command structure designed to be implemented in computer program code is unprotectable under § 102(b). In *Oracle*, the relevant

---

213. *Borland* had been cited 133 times as of July 6, 2015, in the Lexis database. Of these, 115 were positive citations; only 6 either distinguished or criticized the ruling. *Borland* has been cited at least once in every circuit.

214. *See, e.g.*, Hutchins v. Zoll Med. Corp., 492 F.3d 1377, 1383–85 (Fed. Cir. 2007); *Incredible Technologies*, 400 F.3d at 1012; *Mitek*, 89 F.3d at 1556–57; Wyatt Tech. Corp. v. Malvern Instr. Inc., 2009 U.S. Dist. LEXIS 66097, at *6 (C.D. Cal. July 29, 2009); Jamison Bus. Sys., Inc. v. Unique Software Support Corp., 2005 WL 1262095, at *12–13 (E.D.N.Y. May 26, 2005); *see also* Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 538 (6th Cir. 2004) (citing *Borland* approvingly); *O.P. Solutions*, 1999 WL 47191, at *8 (citing *Borland* approvingly).

215. *See, e.g.*, Maddog Software, Inc. v. Sklader, 382 F. Supp. 2d 268, 278–82 (D. N.H. 2005) (applying *Altai* filtration in software copyright case).

216. *See, e.g.*, Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1371–72 (10th Cir. 1997) (rejecting § 102(b) defense based on *Borland*, but affirming non-infringement ruling because command codes were either unoriginal or unprotectable as scenes a faire). Most commentators, however, have cited to *Borland* approvingly. *See, e.g.*, Christina Bohannan, *Reclaiming Copyright*, 23 CARDOZO ARTS & ENT. L.J. 567, 592–93 (2006); Burk, *supra* note 206, at 591–92; Thomas F. Cotter, *The Procompetitive Interest in Intellectual Property Law*, 48 WM. & MARY L. REV. 483, 510 n.115 (2006); Stacey L. Dogan & Joseph P. Liu, *Copyright Law and Subject Matter Specificity: The Case of Computer Software*, 61 N.Y.U. ANN. SURV. AM. L. 203, 211–12 (2005); Herbert Hovenkamp, *Response: Markets in IP and Antitrust*, 100 GEO. L.J. 2133, 2144 n.54 (2012); Dennis S. Karjala, *A Coherent Theory for the Copyright Protection of Computer Software and Recent Judicial Interpretations*, 66 U. CIN. L. REV. 53, 105–07 (1997); Peter Lee, *The Evolution of Intellectual Infrastructure*, 83 WASH. L. REV. 39, 84–85 (2008); Aaron K. Perzanowski, *Rethinking Anticircumvention's Interoperability Policy*, 42 U.C. DAVIS L. REV. 1549, 1563 n.39 (2009); Michael Risch, *How Can* Whelan v. Jaslow *and* Lotus v. Borland *Both Be Right? Reexamining the Economics of Computer Software Reuse*, J. MARSHALL J. COMPUTER & INFO. L. 511, 545–46 (1999).

217. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 990–91 (N.D. Cal. 2012).

218. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1364–68 (Fed. Cir. 2014). The CAFC distinguished *Borland* because that defendant had not literally copied any Lotus code, whereas in the CAFC's view, Google had literally infringed Oracle code. *Id*. at 1365. However, it disagreed with *Borland*'s conclusion that creative methods of operation were ineligible for copyright protection. *Id*. at 1366–67. The CAFC's decision is, in this respect, at odds with Ninth Circuit precedents. *See, e.g.*, Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032, 1035–44 (9th Cir. 2015) (recognizing that Bikram's selection and arrangement of yoga poses and breathing exercises was creative, but rejecting its copyright claims because the poses and exercises were constituent elements of an unprotectable method or system).

command structure drew upon 37 of the 166 Java API packages that Google incorporated into Android.

Some technical background is necessary to inform the legal analysis. The Java API is tightly organized in a set of 166 "packages."[219] Each package identifies types of functions which specific elements of the Java API make available to programmers.[220] The engineers who developed the Java API assigned a specific name to each package. Within each package are numerous related classes of functions. Each class consists of numerous variables and method headers (sometimes called declarations) that specify the type of subroutine to be carried out when the command for that function is invoked.[221] The 37 Java API packages at issue in *Oracle* consisted of 600+ classes, and 6000+ method headers. Google believed that these classes and method headers were unprotectable elements of the Java API,

---

219. The District Court made extensive findings of fact about the Java API packages at issue. *See Oracle*, 872 F. Supp. 2d at 977–83. It indicated that all of the "declarative fact statements set forth in the order are factual findings." *Id*. at 977 n.3.

220. For example, java.math is the package for programming arithmetic functions, whereas java.awt.font is the package for classes and interfaces for fonts. For short descriptions of the Java API packages, see https://docs.oracle.com/javase/1.5.0/docs/api/overview-summary.html [https://perma.cc/EU3L-39HA].

221. For short descriptions of the functionally related sets of classes and declarations within packages, see http://docs.oracle.com/javase/1.5.0/docs/guide/ [https://perma.cc/T9LC-G8R4].

The command structure of the Java API has a standardized format: package. Class.method. An example method header is:

```
package java.lang;

public class Math {
 public static int max(int x, int y);
}
```

This specification identifies the Java command **java.lang.Math.max**, which calls for carrying out the mathematical operation of comparing two numbers and returning the larger of the two. The following line of code uses (or "invokes") that method for computing the maximum of the values stored in variables i and j, storing the resulting value in variable k:

```
k = java.lang.Math.max(i, j);
```

This command may be abbreviated:

```
k = Math.max(i, j);
```

or even further abbreviated as: max().

contending, as Borland had before it, that its reimplementation of these unprotectable elements in independently written code did not infringe.[222]

The District Court recognized that "[t]he overall name tree [of the Java API packages] has creative elements," but it was also "a precise command structure—a utilitarian and functional set of symbols, each to carry out a preassigned function."[223] Because of this, the court concluded that "[t]his command structure is a system or method of operation under Section 102(b) of the Copyright Act and, therefore, cannot be copyrighted."[224] It perceived its ruling to be consistent with the First Circuit's decision that the command structure of Lotus 1-2-3 was uncopyrightable under § 102(b).[225]

The District Court made a finding that compatibility considerations explained why Google had drawn upon the Java API for the Android platform: "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java. Code already written in the Java language would, to this extent, run on Android and thus achieve a degree of interoperability."[226] Java programmers, the court noted, had written millions of lines of code using these method headers and classes, and reuse of that code on the Android platform required use of the same method headers.[227] Although Google copied the exact names and functions of the Java API classes and method headers, the court noted that Google "took care to use different code to implement the six thousand-plus subroutines (methods) and six-hundred-plus classes."[228]

Oracle's appeal found a very receptive audience in the CAFC. There were three main bases for the CAFC's decision. One concerned the "copyrightability" issue in the case. The CAFC agreed with Oracle that the API packages at issue were protectable by copyright law under § 102(a)

222. *Oracle*, 872 F. Supp. 2d at 998.

223. *Id*. at 976–77.

224. *Id*. at 977.

225. *Id*. at 990–91. While concluding that § 102(b) provided a sound basis for its ruling in Google's favor, the District Court proffered the merger doctrine as an alternative ground because "[u]nder the rules of Java, [the method headers] *must be identical* to declare a method specifying the *same* functionality." *Id*. at 976. The merger doctrine is discussed in Part IV.

226. *Id*. at 978. The District Court also held that the names of the individual commands were unprotectable under the copyright doctrine that words and short phrases cannot be copyrighted. *Id*. at 997. Although I will not address this aspect of the District Court's ruling, it is not implausible given judicial receptivity to this defense in rulings such as *Southco, Inc. v. Kanebridge Corp. See* 390 F.3d 276, 285 (3d Cir. 2004).

227. *Oracle*, 872 F. Supp. 2d at 1000.

228. *Id*. at 977.

because Sun's engineers had been highly creative in designing the Java API, including in naming and organizing the method headers and the classes of the API packages.[229] Hence, copyright's originality standard was easily met and the resulting expression was copyright-protectable. The CAFC thought that the protection conferred by § 102(a) could not be taken away by § 102(b).[230] As in *Whelan*, the CAFC perceived § 102(b) to be a restatement of the idea/expression distinction, which had no pertinence in a case involving a highly detailed structure, such as the Java API packages.[231]

However, the CAFC did not correctly understand the copyrightability issue in the case. Courts use that term in three different ways: sometimes they use it to indicate that a particular work is (or is not) proper subject matter for copyright protection (e.g., a literary work),[232] sometimes to indicate that the originality and fixation requirements have (or have not) been met as to a particular work,[233] and sometimes to indicate that the aspect of a work that the defendant copied was outside the scope of copyright protection available to that work.[234]

The CAFC seems to have regarded the District Court as having used the term in the first or second sense, which explains why it regarded Google's § 102(b) defense as calling into question the copyrightability of all program code.[235] However, the District Court in *Oracle*, like the First Circuit in *Borland*, used the term "copyrightability" in the third sense of the term. The District Court did not question the copyrightability of the work at issue in the case, namely, the Java Special Edition 5.0 document from which Google drew the 37 API packages.[236] Rather, the District Court understood Oracle

229. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1356 (Fed. Cir. 2014).

230. *Id*. at 1556–57.

231. *Id*. at 1367–68.

232. *See, e.g*., Kelley v. City of Chicago, 635 F.3d 290, 306 (7th Cir. 2011) (gardens not copyrightable subject matter).

233. *See, e.g.*, Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 362–63 (white pages listings of telephone directories are uncopyrightable because they lack sufficient originality to support copyright).

234. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996) (considering whether the Lotus command hierarchy was "copyrightable" part of the Lotus program).

235. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1379–81 (Fed. Cir. 2014).

236. The work of authorship at issue in *Oracle* was not, in fact, a computer program, as the CAFC opinion seems to imply, *id.* at 1355–56, but was a textual document entitled Java Special Edition 5.0, which sets forth the component parts of the Java API. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 979 (N.D. Cal. 2012). The Java Platform Special Edition document sets forth the specification of the Java API as well as providing other pertinent Java platform information. The current version of this document can be found at

to be claiming that Google's use of the 6000+ method headers constituted a non-literal infringement of that document.[237] The District Court understood the *Oracle* case to present a scope of protection issue, that is, whether the command structure of these method headers was protectable by copyright law.[238]

Second, the CAFC accepted Oracle's claim on appeal that Google literally copied 7000 lines of Oracle source code, as well as nonliterally copied the SSO of the Java API classes.[239] However, this directly contradicted the District Court's factual finding that there was no literal copying of program code. "[A]ll agreed," it said, "that Google had not literally copied the software but had instead come up with its own implementations of the 37 API packages."[240] The CAFC did not understand that the API elements that Google used in Android were not in themselves software; they were specifications that identified functions that the program was designed to perform. In order to write programs to implement these functions in accordance with rules set forth in the Java API, the source code must include the API method headers (or declarations); those method headers, however, do not become part of the executable code.[241] The CAFC's misperception about the literal or nonliteral copying issue reinforced its conception that Google's defense would undermine software copyright protection.

Third, the CAFC disagreed strongly with the District Court about Google's compatibility defense. The CAFC categorically denied that the case law had recognized a "compatibility exception" to copyrightability of

---

https://docs.oracle.com/javase/8/ [https://perma.cc/4AUB-JAVN]. This document can be downloaded for free from a public website.

237. *Id*. at 975 (all agreed that Google had not literally copied Oracle software).

238. *Id*. at 975–76.

239. *Oracle*, 750 F.3d at 1361. Oracle should not have been able to change its theory of the facts on appeal to make Google's defense seem weaker. Had the Supreme Court taken the case, *Oracle* should have been treated as a nonliteral infringement case, not a literal copying case.

240. *Oracle*, 872 F. Supp. 2d at 975. Under the Supreme Court's recent decision in Teva Pharm. v. Sandoz, 574 U.S. ___, 135 S. Ct. 831 (2015), the CAFC should have deferred to lower court findings of fact and reversed them only when the findings were clearly in error.

241. Like the District Court in *Oracle*, one post-*Oracle* decision has distinguished declaring code (i.e., method headers), the function of which was to call for the performance of a particular function, and implementing code, the function of which was to be the object code to carry out that function. SAS Inst., Inc. v. World Programming Ltd., 64 F. Supp. 3d 755, 777 (E.D.N.C. 2014).

program APIs.[242] Echoing *Franklin*, the CAFC characterized compatibility "as a commercial and competitive objective," which had no bearing on copyrightability.[243] In addition, it perceived no evidence that the design of the Java API packages were constrained by "compatibility requirements" of pre-existing programs, the only constraints it thought might have significance.[244] The CAFC was also skeptical of the defense, saying "Google designed Android so that it would *not* be compatible with the Java platform,"[245] thereby defeating the Java goal of allowing programmers to "write once, [code that will] run everywhere."[246] This frontal attack, not only on Google's compatibility defense in *Oracle*, but on compatibility defenses in software copyright cases, requires a more elaborate response.

D. THE IMPLICATIONS OF § 102(b) FOR COMPATIBILITY DEFENSES

The functionality of computer program APIs is unquestionable insofar as programs can interoperate with existing programs only if they conform to the interface procedures that existing programs have adopted. This explains why the Ninth Circuit has twice unequivocally affirmed that APIs, insofar as they constitute the functional requirements for achieving compatibility with other programs, are unprotectable procedures under § 102(b).[247] The Ninth Circuit has thus recognized in these and subsequent cases that § 102(b) is more than merely a restatement of the idea/expression distinction,[248] a proposition that the CAFC refused to accept in its *Oracle* decision.[249] For more than two decades, courts have consistently followed

242. *Oracle*, 750 F.3d at 1368–71. Yet, in *Atari Games*, the CAFC intimated that no infringement might have been found if AG had copied only what was necessary for interoperability. 975 F.2d 832, 844–45 (Fed. Cir. 1992).

243. *Oracle*, 750 F.3d at 1371 (quoting Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983)).

244. *Id*.

245. *Id*.

246. *Id*. at 1348.

247. *See* Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602–05 (9th Cir. 2000); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992).

248. *See also Bikram's Yoga*, 803 F.3d at 1041 ("That the sequence may possess many constituent parts does not transform it into a proper subject matter of copyright protection. Virtually any process or system could be dissected in a similar fashion."). Defendants in the *Oracle*, *Cisco v. Arista* and *Synopsys v. ATopTech* cases tried to persuade the Ninth Circuit to amend its *Bikram's Yoga* decision to clarify that the CAFC has misunderstood Ninth Circuit law concerning § 102(b). *See* Motion for Leave to File Brief Amici Curiae of Google Inc., Arista Networks, & ATopTech, Inc., and Brief Amicus Curiae of Google Inc., Arista Networks, & ATopTech, Inc., at 3–12, Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032 (2015) (No. 13-55763).

249. *See Oracle*, 750 F.3d at 1365–67. The CAFC has previously recognized that § 102(b) has broader significance. *See* Hutchins v. Zoll Med. Corp., 492 F.3d 1377, 1383

the rulings in *Sega* and *Altai* that program interfaces necessary for interoperability are unprotectable by copyright law,[250] making the CAFC's *Oracle* decision an outlier. Although the CAFC recognized in *Oracle* that it should apply Ninth Circuit case law,[251] and even purported to do so,[252] its ruling is at odds with Ninth Circuit precedents. Because the Supreme Court decided not to review the CAFC ruling,[253] software developers now face an uncertain future when they raise compatibility defenses.[254]

The CAFC's *Oracle* decision is most worrisome in its outright repudiation of software compatibility defenses, its resurrection of the anti-compatibility *Franklin* dictum, and its refusal to acknowledge that the Ninth Circuit in *Sega* and *Connectix* treated program interfaces as unprotectable procedures under § 102(b). As amicus curiae briefs of computer scientists, software companies, industry associations, and public interest groups filed in support of Google's petition for certiorari attest,[255] freedom to reuse APIs, insofar as they are necessary for interoperability, promotes healthy competition and ongoing innovation in the software industry. More than two decades of copyright rulings have endorsed this freedom, and the software industry has flourished under this legal regime.[256] While one can

(Fed. Cir. 2007) ("[C]opyright protection does not extend to the methods that are performed with program guidance.").

250. See *infra* text accompanying notes 259–64 for a review of those cases.

251. Oracle, 750 F.3d at 1353. Oracle's appeal of the District Court's copyright ruling went to the CAFC, instead of the Ninth Circuit, because Oracle had alleged patent, as well as copyright, infringement. *Id*.

252. *See* 2 PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT, § 2.15.2.1 (3d ed. 2005 & Supp. 2016) (arguing CAFC purported to follow Ninth Circuit law in *Oracle*). The CAFC disingenuously interpreted *Sega* as having considered compatibility as a factor in fair use analysis, choosing to ignore the court's unequivocal statements about interface procedures being excluded from copyright under § 102(b). *See Oracle*, 750 F.3d at 1369–70. The CAFC said that compatibility could be considered in connection with Google's fair use defense on remand. *Id*. at 1372–77. In rejecting the District Court's copyrightability ruling, the CAFC relied on its decision in Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572 (Fed. Cir. 1990) (also rejecting a software compatibility defense), which the CAFC regards as a correct interpretation of Ninth Circuit law. *Oracle*, 750 F.3d at 1353–61.

253. Google Inc. v. Oracle Am., 135 S. Ct. 1021 (2015).

254. At risk are compatibility defenses in the *Cisco v. Arista* and *Synopsys v. ATopTech* cases because the plaintiffs included a patent claim in the case, meaning any appeal will go to the CAFC. *See supra* note 30.

255. Briefs available online at *Google Inc. vs. Oracle America, Inc.*, SCOTUSBLOG, http://www.scotusblog.com/case-files/cases/google-inc-v-oracle-america-inc/ [https://perma.cc/XC5H-78ZL].

256. *See, e.g.*, Brief Amicus Curiae of Computer and Communications Industry Ass'n in Support of Petitioner at 4–8, *Oracle*, 135 S. Ct. 1021 (2015) (No. 14-410), 2014 WL 5868946 (emphasizing importance of pro-compatibility decisions in fostering competition and innovation in the software industry).

hope that courts in the future will reject the CAFC's analysis in *Oracle*, or distinguish *Oracle* from other compatibility cases,[257] that decision has reopened a longstanding debate about the scope of copyright in computer programs, about the implications of § 102(b), as well as about compatibility defenses. The CAFC would shunt compatibility considerations away from § 102(b) and merger defenses, leaving them to the case-by-case vagaries of fair use.[258]

Before attempting to grapple with the different perspectives in the District Court and CAFC's interpretations of Google's compatibility and § 102(b) defenses, it is useful to recognize that several different types of interoperability issues have arisen in prior cases.

1. Competing Applications' Compatibility with the Same Operating Systems: In *Altai*, the litigants were competitors in the market for scheduling programs designed to interoperate with IBM operating system programs.[259] Similarities in the parameter lists for these programs were largely due to the need to conform to IBM's interface procedures.

2. Unlicensed Application Developer Compatibility with a Popular Platform: In *Sega,* Accolade wanted to adapt an existing videogame program so it would run on the Sega Genesis platform.[260]

3. Emulation Software to Enable Applications Developed for a Popular Platform to Run on an Alternative Platform: Connectix developed a program to emulate the functionality of the Sony PlayStation

257. The only post-*Oracle* decision to address the substance of the CAFC's ruling so far was SAS Inst., Inc. v. World Programming Ltd., 64 F. Supp. 3d 755, 777 (E.D.N.C. 2014). SAS argued that the CAFC's ruling undermined WPL's compatibility defense. WPL developed software that emulated the functionality of the SAS statistical analysis program so that users of the SAS program could switch to its program and still reuse the scripts (mini-programs) they had constructed in the SAS language. The court distinguished the CAFC's ruling, saying that WPL had only used the SAS language, which under the *Oracle* decision, all were free to use. *Id*. at 776–78.

258. *See Oracle*, 750 F.3d at 1371–77. The CAFC remanded the case for retrial on the fair use issue. *Id*. at 1376–77. See *infra* notes 363, 455 and accompanying text for a discussion of compatibility issues in the context of fair use.

259. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 714–15 (2d Cir. 1992).

260. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514–15 (9th Cir. 1992). Similar to *Sega* was Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832 (Fed. Cir. 1992). Like Accolade, Atari Games (AG) sought to develop videogames that would run on a popular platform. *Id*. at 836–37. The CAFC rejected AG's interoperability defense because AG had copied more from Nintendo's program than was necessary to achieve interoperability. *Id*. at 845.

platform, so that games developed for that platform could be played on the defendant's virtual machine.[261]

4. Emulation of a Feature to Enable Prospective Customers' Mini-Programs Created in One Program to Run on a Competing Program: Borland created an emulation user interface to enable interoperability with user-created macros.[262]

5. Development of a New Operating System to Enable Continued Use of an Application Program: In *Bateman v. Mnemonics, Inc*., the defendants had to develop their own operating system to interoperate with an application program they had developed for their business before Bateman terminated their license to use his OS to run the application.[263]

6. Reuse of Software to Achieve Compatibility with Hardware: In *Lexmark*, Static Control developed chips loaded with a copy of a Lexmark program so that Lexmark's competitors could make printer cartridges that would successfully interoperate with Lexmark printers.[264]

Despite the factual and contextual differences among these cases, the courts in each held that the interfaces necessary for achieving technical interoperability were unprotectable elements of copyrighted software. The CAFC in *Oracle*, however, refused to acknowledge this.[265] These six types

261. Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000). *Connectix* was like *Franklin*, except that Connectix reimplemented the interface in non-infringing code instead of copying the platform code bit for bit, as Franklin had. *Id.* at 601.

262. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 810 (1st Cir. 1995). Similar to *Borland* was SAS Inst., Inc. v. World Programming Ltd., 2013 EWCA Civ. 1482 (UK Ct. Ap. 2013) (not infringement to emulate the interface and functionality of SAS software to enable users to port over to another platform the mini-programs they had constructed in the SAS language). *See also SAS v. WPL*, 64 F. Supp. 3d at 775–78; Bay State Tech., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1088 (D. Mass. 1996) (not infringement for CAD software competitor to develop data translator software to allow files created in plaintiff's program to be "read" in the defendant's program).

263. Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1536–40 (11th Cir. 1996). The defendants reverse-engineered Bateman's OS to discern elements needed to achieve compatibility, which they claimed required copying of some literal code. *Id*. at 1539 n.18, 1545.

264. Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 530–31 (6th Cir. 2004); *see also Secure Services*, 722 F. Supp. at 1356–64 (not infringement to reimplement competitor's variation of a standard protocol so new entrant could sell interoperable secure fax machines to the U.S. government and its contractors); NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d 1177, 1188 (N.D. Cal. 1989) (rejecting Intel's claim of infringement of microcode that was necessary to enable NEC's hardware to be compatible with Intel's hardware). *But see* Compaq v. Procomm, 908 F. Supp. 1409 (S.D. Tex. 1995) (parameters and their sequence not protectable, but preliminary injunction issued because defendant copied more than was necessary to achieve compatibility with Compaq hard drives).

265. *Oracle*, 750 F.3d at 1370–71.

of technical compatibility cases might seem to suggest that APIs, by their nature, should be considered unprotectable elements of computer programs. The District Court in *Oracle* did not interpret these precedents so broadly; indeed, it rejected Google's legal argument to this effect earlier in the case.[266] The District Court regarded its holding as a narrow one: the Java API elements at issue in *Oracle* were unprotectable methods or systems under § 102(b).[267] The law is unsettled as to whether all interfaces, APIs, and command structures should be treated the same.[268] The terms "interface," "APIs," and "interoperability" unfortunately do not have fixed and unalterable meanings.[269]

The District Court regarded Google's interest in enabling technical compatibility of Java programs with the Android platform as a legitimate explanation for Google's use of the Java API packages. It observed, for instance, that "millions of lines of code had been written in Java before Android arrived," and all of them had "necessarily used the java.package.Class.method( ) command format."[270] These programs had been developed and were owned by firms other than Google.

> In order for at least some of this code to run on Android, Google was required to provide the same java.package.Class.method( ) command system using the same names with the same 'taxonomy' and with the same functional specifications.[271]

Hence, it was not just Google that was interested in enabling Java programs to run on Android; the developers of existing programs would want them to run (or be easily adapted to run) on Android too. The District Court found that Google had used "what was necessary to achieve a degree of interoperability—but no more, taking care, as said before, to provide its own implementations."[272] In this respect, the District Court thought *Oracle*

266. Oracle Am. Inc. v. Google Inc., 810 F. Supp. 2d 1002 (N.D. Cal. 2011).

267. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 1000–01 (N.D. Cal. 2012).

268. *Bateman*, 79 F.3d at 1547 (rejecting argument that interface specifications are uncopyrightable as a matter of law). A document that sets forth the component elements of an API with original commentary to aid users would be copyrightable, but the copyright in that document should not be infringed insofar as reuse of the specified API is necessary to achieve program interoperability.

269. *See, e.g.*, BAND & KATOH, INTERFACES 2.0, *supra* note 10, at 41; *see generally* ASHWIN VAN ROOIJEN, THE SOFTWARE INTERFACE BETWEEN COPYRIGHT AND COMPETITION LAW: A LEGAL ANALYSIS OF INTEROPERABILITY IN COMPUTER PROGRAMS (2010).

270. *Oracle*, 872 F. Supp. 2d at 1000.

271. *Id.*

272. *Id.*

was analogous to the *Sega* and *Connectix* cases.[273] *Connectix* seemed especially relevant because that defendant, like Google, had achieved only partial interoperability.[274]

The CAFC directly challenged Google's interoperability claim, saying, "Google designed Android so that it would *not* be compatible with the Java platform."[275] There was, in its view, "no evidence in the record that any [fully Java-compatible] app [running on Android] exists and [the trial court] points to no Java apps that either pre-dated or post-dated Android that could run on the Android platform."[276] The CAFC should, however, have either deferred to the District Court's factual finding that millions of lines of Java code could run on Android, even if that court did not identify specific programs, or have sent the case back to the District Court for further proceedings to develop a fuller record about compatibility issues.[277]

The CAFC asserted that Google adopted the 37 API packages for Android "to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue."[278] That is, Google's use of Java API method headers would make it easier for Google to attract Java developers to create apps for the Android platform. The CAFC seemed to perceive this as unfair free-riding because it would "accelerat[e]" Google's development process "by 'leverag[ing] Java for its existing base of developers.'"[279] The CAFC may also have been swayed by the existence of a licensing program that Oracle (and its predecessor Sun) had established to ensure that Java compatibility goals would be maintained, which Google arguably bypassed by using Java packages in Android without a license.[280] Oracle claimed that Google's success with

273. *Id*.

274. *Id*. at 1000–01 (noting that Connectix implemented only 137 of 242 Sony BIOS functions).

275. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014).

276. *Id*.

277. *See supra* note 240 (concerning the deference issue). The CAFC was incorrect in asserting that no pre-existing Java programs can run on the Android platform.

278. *Oracle,* 750 F.3d at 1371. The CAFC also rejected Google's argument that the Java API command structure had become unprotectable because it was an industry standard. *Id*. at 1372. Google had made this claim below, but failed to present evidence to support it. *Oracle*, 872 F. Supp. 2d at 999 n.9.

279. *Oracle*, 750 F.3d at 1371.

280. *See id.* at 1350. Google had negotiated with Sun for a time to license the Java technology, but thought it did not need a license to use only parts of the API and not the suite of Java technologies that Sun licensed to some developers. The failure to license the API did not cut against Accolade in *Sega*. *See* 977 F.2d 1510, 1514 (9th Cir. 1992).

Android had "fragment[ed]" Java, defeating its purpose, and thwarting Oracle's ability to develop its own mobile platform.[281]

Let us accept that Google's motivation in adopting those 37 Java API packages was, in no small part, to facilitate a kind of human interoperability. Oracle has estimated that there are roughly nine million Java programmers in the world.[282] All have made significant investments in learning the rules and syntax of Java, as well as in using the method headers and classes of Java API packages, and implementing the Java API in programs that run on a wide variety of machines to accomplish a wide range of tasks. They are familiar with Java commands and command structures. When they write code, they express themselves with Java commands. If Judge Boudin was right to consider user investments in learning the Lotus command structure and constructing macros in the Lotus language as supporting Borland's right to reimplement the Lotus command hierarchy in QP,[283] other courts should be receptive to user investments in learning to use Java as a consideration that weighs in favor of limiting the scope of copyright in cases such as *Oracle*.

It is understandable that Java programmers want to reuse command names to identify specific functions that their programs are designed to perform. Naming commands is difficult because

> [e]ach name must succinctly and intuitively capture the role and function of the concept being named. Because programs use more names than can be reasonably remembered, the names must be

281. These points were heavily emphasized in Oracle's brief to the CAFC. *See* Opening Brief and Addendum of Plaintiff-Appellant [hereinafter Opening Brief] at 13–29, 51–55, *Oracle*, 750 F.3d 1339 (No. 13-1021), 2013 WL 518611. Sun had, in fact, developed Java Micro Edition platforms for mobile devices that were not fully compatible with the Java language specification. Java MEs have been used on a large number of mobile devices. These platforms did not achieve as much success as Android, in part because of their use of "draconian" security measures and putting interests of telco carriers above interests of users. *See* Matthew Powell, *Why did Nokia fail to compete with Samsung, Apple etc., despite being the giant of the mobile phone industry?*, QUORA (Feb. 15, 2015), http://www.quora.com/Why-did-Nokia-fail-to-compete-with-Samsung-Apple-etc-despite-being-the-giant-of-the-mobile-phone-industry [https://perma.cc/493C-BHPV].

282. Nikita Salnikov-Tarnovski, *How Many Java Developers Are There in the World?*, DZONE (July 20, 2012), https://dzone.com/articles/how-many-java-developers-are [https://perma.cc/GVL8-GVRV].

283. *See supra* text accompanying notes 205–208; *see also* John Bergmayer, *Compatibility is About Competition, Too*, PUBLIC KNOWLEDGE (Feb. 26, 2015), https://www.publicknowledge.org/news-blog/compatibility-is-about-competition-too [https://perma.cc/Z9FY-J943] ("It's about allowing developers to make the most of their skills and *their* code, not about Android trying to take something away from Oracle's version of Java.").

> systematized to be easy to reconstruct and interpret later. This, in turn, will affect how easily you or others can understand, fix, and extend that same code months or years later. As a result, bad names create buggy code, and good names can deliver tremendous boosts to team productivity.[284]

The Java API systematized names in a very efficient way, which is why the language and the API has been so widely used. The creators of Java can, of course, take some credit for this success, but they cannot take all of it, just as they cannot claim ownership over consumer choice and investment in determining the popularity of any given product.[285]

Oracle contended that Google could have used different names for method headers for specific functions in support of its claim that Google did not have to copy the method headers that Sun's engineers developed.[286] In effect, Oracle was arguing that Google should have forced Java programmers to learn a new dialect of Java to write apps for the Android platform. Yet, had Google developed a new Java dialect, Google would have confused Java programmers who would have had to relearn to program in Java in a different way for Android than for other devices. This would have caused a much more serious fragmentation of Java than Google caused by adopting only 37 of the Java packages instead of all of them. So if Oracle really valued the integrity of Java, it would not have wanted Google to develop alternative method headers in the Java language.

The best way to have avoided the fragmentation of Java, of which Oracle complained, would have been for Google to adopt all 166 of the API packages instead of just 37 of them. Had Google adopted all 166 packages, a more complete interoperability of Java programs on Android would presumably have been achieved. It would also have made the *Oracle* case more like *Sega*.

The Java API may be a much more complex procedure than the Genesis interface at issue in *Sega*, but the complexity of a procedure does not change its essential character. Under Ninth Circuit precedents,[287] the Java API and its constituent parts would be unprotectable under § 102(b) insofar as its use

---

284. Excerpt from an online discussion in answer to the question "Why is naming things hard in computer science and how can it be made easier?", QUORA (answer updated Feb. 17, 2014), http://quora.com/Why-is-naming-things-hard-in-computer-science-and-how-can-it-be-made-easier [https://perma.cc/AW92-BEEZ].

285. *See generally* Michal Shur-Ofry, *Popularity as a Factor in Copyright Law*, 59 U. TORONTO L.J. 525 (2009).

286. *See* Opening Brief, *supra* note 281, at 32–33, 51–52.

287. *See* Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602–03 (9th Cir. 2000); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992).

was needed to achieve interoperability, as the District Court in *Oracle* held.[288]

Even if the Java API as a whole might be an unprotectable system, method, or procedure under Ninth Circuit precedents, it is unclear whether or why the structure of 37 of the Java API packages should be regarded as constituting a system or method of operation. The District Court did not explain why a subset of the API should be regarded as a system or method as well.

Yet, if we accept that the Java API as a whole was an unprotectable system or procedure, it would be logical to conclude that its constituent parts should be unprotectable as well. The Ninth Circuit did not treat Connectix's partial implementation of the Sony PlayStation interface as any less of a procedure than in *Sega*.[289] Consider also that a second comer would be free to copy 6000 items from an uncopyrightable compilation of 60,000 white pages listings of a telephone directory. Baker did not copy Selden's forms exactly, but rather adapted the forms so that it would be easier for accountants to enter data on an ongoing basis.[290] Baker thus used some of the Selden system in his forms, but he did not become an infringer for not using all of it. Consider further that Android is a special purpose platform for mobile devices, a type of platform that was not in contemplation when Java was developed. The success of Java has mainly been achieved with server-side systems and large-scale enterprise software that can run on multiple machines. It stands to reason that a special purpose platform for mobile devices would need to have some of the same, but also some different, functionalities that its API would need to accommodate. This could explain why Google used only some, but not all, of the Java API

288. *Oracle*, 872 F. Supp. 2d at 997–1000; *see* Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990) ("When the questions on appeal involve law and precedent on subjects not exclusively assigned to the Federal Circuit, the court applies the law which would be applied by the regional circuit.") (quoted in *Oracle*, 750 F.3d at 1353). Yet, the CAFC cited to its opinions in *Atari Games* ten times as the most relevant "Ninth Circuit" precedents. It cited the *Apple v. Microsoft* decision only once for a minor point, even though that decision is the Ninth Circuit's principal software copyright decision. The Ninth Circuit has cited the *Atari Games* decision only twice in passing in the last decade in the Westlaw database, whereas it has cited *Apple v. Microsoft* eighty-one times.

289. *See Connectix*, 203 F.3d at 599, 609 (noting lack of full compatibility), 602–03, 607 (Sony interface held to be § 102(b) procedure); *see also* Google Appellee Brief, *supra* note 212, at 37 (noting that Connectix had used 137 of 242 elements of the Sony PlayStation interface).

290. Samuelson, *Baker v. Selden*, *supra* note 155, at 164.

packages.[291] Even if avoiding fragmentation of Java was a commercial and competitive objective for Oracle, that consideration has no bearing on whether the Java API packages were unprotectable under § 102(b).

Until the *Oracle* decision, the law was settled that program interfaces necessary to achieving interoperability among programs or with hardware were unprotectable by copyright law because of their inherent functionality. The *Oracle* decision incorrectly interpreted Ninth Circuit precedents, including the significance of § 102(b) in cases involving program APIs. Healthy competition and innovation in the software industry has depended for decades on the ability of second comers to build new programs that interoperate with older ones, even when the initial developer of the API at issue has not consented to this use.

IV. FUNCTIONALITY AND EXPRESSION SOMETIMES MERGE IN SOFTWARE CASES

Copyright law has long recognized that when there is only one or a small number of ways to express an idea or function, copyright protection will be withheld to any expression that has merged with the idea or function.[292] Software APIs necessary for achieving interoperability are among the computer program innovations in which function and expression effectively merge. APIs specify a system of rules and procedures to which other programs must strictly conform in order to attain compatibility with an existing program. The fact that software engineers might ex ante have had numerous choices in how to design an API does not make the API copyrightable because that API significantly constrains design choices of subsequent programmers.

Section A discusses the origins and scope of copyright's merger doctrine. Section B considers the role of the merger doctrine in architectural work and software cases. Section C reviews software copyright cases in which a first programmer's design choices constrained the design choices of subsequent programmers, particularly when reuse of an API is necessary to achieving interoperability. Section D recognizes that on some occasions, function and expression may merge as to program code. Section E identifies several errors in the CAFC's interpretation of the merger doctrine in its *Oracle* decision.

---

291. It is common for technologists to tinker with existing technologies and adapt them to different uses than their manufacturers intended. Considerable innovation has occurred from these adaptations. *See generally* ERIC VON HIPPEL, DEMOCRATIZING INNOVATION (2005).

292. *See infra* text accompanying notes 293–301; *see also* Pamela Samuelson, *Reconceptualizing Copyright's Merger Doctrine*, 63 J. COP. SOC'Y U.S.A. 417, 417 (2016).

A. ORIGINS OF THE MERGER DOCTRINE

The challenge of distinguishing expression and functionality in copyrighted works has been with U.S. copyright law for a very long time. The Supreme Court first addressed the issue in *Baker* when holding that Selden's copyright extended to his *explanation* of his bookkeeping system, not to the bookkeeping *system* itself.[293] The Court characterized Selden's forms as "necessary incidents" to this useful art, and since practicing that system required use of Selden's forms, the forms were uncopyrightable as well.[294]

Courts and commentators often point to the "necessary incidents" language when speaking of *Baker* as having originated copyright's merger doctrine.[295] The selection and arrangement of columns and headings in Selden's forms had, in this conception of *Baker*, in effect, merged with the system and the method of operation that Selden anticipated accountants would carry out when using the forms.[296]

---

293. 101 U.S. 99, 101 (1879). *Baker* is best understood as holding that systems, methods of operation, and other useful arts are excluded from the scope of copyright protection, a doctrine that is now codified in § 102(b). *See* Samuelson, *Why Copyright Excludes Systems*, *supra* note 124, at 1931–42 (explaining *Baker* and its progeny on this point). Contrary to common perception, *Baker* is not the source, or even as a classic statement, of the idea/expression distinction, although many courts characterize in this way. *See id*. at 1924–36 (explaining that the exclusion of ideas from the scope of copyright predated *Baker* and was not at issue in that case).

294. *Baker*, 101 U.S. at 103. The Nimmer treatise has asserted that Baker's forms did not infringe because they differed in some respects from the Selden forms. NIMMER ON COPYRIGHT, *supra* note 10, at § 2.18[B][1]. The relevant forms can be found in Samuelson, *Baker v. Selden*, *supra* note 155, at 170–71. The logic of the Court's decision, however, supports the view that exact copying of the forms would not infringe. *See, e.g.*, BENJAMIN KAPLAN, AN UNHURRIED VIEW OF COPYRIGHT 63–64 (1966); HORACE G. BALL, THE LAW OF COPYRIGHT AND LITERARY PROPERTY 111–12, 125–28 (1944) (discussing *Baker* and its progeny as precedents for the unprotectability of systems of business, plans of instruction, or methods of practicing an art or playing a game); ARTHUR W. WEIL, AMERICAN COPYRIGHT LAW 193–94 (1917) (citing *Baker* and its progeny as precluding copyright protection for plans, methods, and arts).

295. Among the many cases that cite to *Baker* as the origin of the merger doctrine is Arica Inst. v. Palmer, 970 F.2d 1067, 1076 (2d Cir. 1992).The Nimmer treatise is among the many sources that so credit *Baker*. NIMMER ON COPYRIGHT, *supra* note 10, at §2.18. I have elsewhere argued that *Baker* did not, in fact, give birth to the merger doctrine, but rather to the exclusion of procedures, processes, systems, and methods of operation now codified in § 102(b). *See* Samuelson, *Reconceptualizing Merger*, *supra* note 292, at 419–22. In numerous cases, merger and § 102(b) defenses have been treated interchangeably. *See id*. at 451–53.

296. *See, e.g.*, Burk, *supra* note 206, at 591 (Selden's "accounting forms were the only way to express the accounting system [so] that the idea and expression had merged.").

After *Baker*, courts in the U.S. have mainly struggled with the problem of the separability or merger of functionality and expression when deciding whether pictorial, graphic, or sculptural (PGS) works are protectable by copyright law.[297] The 1976 Act allows PGS works to be copyrighted insofar as they embody original expression that "merely [] portray[s] the appearance of [an] article or [] convey[s] information."[298] If such works have "an intrinsic utilitarian function" that goes beyond portraying an appearance or conveying information, they are disqualified from U.S. copyright protection.[299] If the design of a PGS work "incorporates pictorial, graphic, or sculptural features that can be identified separately from and are capable of existing independently of, the utilitarian aspects," then the work is copyrightable.[300] If functionality and expression are inseparable (i.e., merged), then no matter how creative its design, the PGS work is unprotectable by copyright law.[301]

Mazer v. Stein is a classic copyright case in which expression and functionality were separable.[302] Stein's statuette of a Balinese dancer was eligible for copyright protection because the statuette was a work of art that existed independently from the lamp.[303] It did not matter that the statuette was being commercially exploited as the base for Stein's lamps because the lamp did not function any better or worse for having the statuette as its base. Jewelry, fabric designs, and dolls are among the other types of intellectual creations that, having satisfied the separability criterion, enjoy copyright protection.[304]

The overall design of a chair or automobile, by contrast, may well have an aesthetic character, but any expressive aspects of these creations cannot be separated from (that is, they are merged with) their utilitarian functions.[305] The merger of functionality and expression in chairs,

297. *See, e.g.*, Kieselstein-Cord v. Accessories by Pearl, Inc., 632 F.2d 989 (2d Cir. 1980) (separable expression in belt jewelry); Carol Barnhart Inc. v. Economy Cover Corp., 773 F.2d 411 (2d Cir. 1985) (no separable expression in mannequin).

298. 17 U.S.C. § 101 (definition of "useful article").

299. *Id.* (definition of "useful article").

300. *Id.* (definition of PGS works).

301. *See, e.g.*, Burk, *supra* note 206, at 591 (characterizing the useful article/PGS rule as a kind of merger doctrine).

302. 347 U.S. 201 (1954).

303. *See id.* at 215–18.

304. *See, e.g.*, Kieselstein-Cord v. Accessories by Pearl, Inc., 632 F.2d 989, 993 (2d Cir. 1980) (finding that the conceptually separable artistic elements of belt buckles are copyrightable).

305. *See* H.R. REP. NO. 94-1476, at 55 (1976) (giving numerous examples of designs of useful articles which, even if original, are unprotectable under U.S. copyright law).

automobiles, and other useful articles results in no copyright protection whatsoever in the U.S.[306] Moreover, to ensure that copyrights in technical drawings or depictions of functional creations do not indirectly undermine the no-copyright-for-functionality rule, the 1976 Act and case law clarify that any copyright in the drawing does not extend protection to the technical content or functional creations depicted therein.[307]

B. THE ROLE OF THE MERGER DOCTRINE IN ARCHITECTURAL WORK AND SOFTWARE CASES

Congress has created two limited exceptions to the no-copyright-for-functional-creations rule: one for computer programs and another for architectural works.[308] By designating programs and architecture as copyrightable subject matters, Congress did not jettison the longstanding rule that copyright does not protect functionality. Instead, it shifted the problem of assessing whether expression and functionality are separable or merged so that the question is, generally speaking, no longer about eligibility for copyright protection,[309] but rather about assessing the proper scope of protection for such works.

The courts have explicitly recognized the relevance of the merger doctrine as a limit on the scope of copyright protection for functional design elements of architectural works. In *Zalewski v. Cicero Builder Dev., Inc.*, for example, the Second Circuit recognized that efficiency considerations may significantly constrain the design of buildings, as well as computer programs, and adapted the filtration factors first articulated in *Altai* to apply

306. *See, e.g.*, Brandir Int'l, Inc. v. Cascade Pacific Lumber Co., 834 F.2d 1142, 1147 (2d Cir. 1987) (ribbon design for bicycle rack held unprotectable by copyright law as useful article lacking separable expression).

307. 17 U.S.C. § 113(b); *see, e.g.*, Niemi v. American Axle Mfg., No. 05-74210, 2006 WL 2077590 (E.D. Mich. July 24, 2006) (unauthorized manufacture of machine did not infringe copyright in drawing); Eliya, Inc. v. Kohl's Dept. Stores, 82 U.S.P.Q.2d 1088 (S.D.N.Y. 2006) (manufacture of shoe did not infringe copyright in drawing); Fulmer v. United States, 103 F. Supp. 1021, 1022 (Ct. Cl. 1952) (copyright in drawing of parachute design did not give its author an exclusive right to make parachutes like that depicted in the drawing).

308. *See supra* note 2 (discussing the Congressional decision to extend copyright protection to programs). Congress added architectural works to copyright subject matter in 1990. Copyright Improvements Act of 1990, Pub. L. No. 101-650, Tit. VII, 104 Stat. 5133 (codified in various sections of 17 U.S.C.). Copyright Office regulations state that structures, such as "bridges, cloverleafs, dams, walkways, tents, recreational vehicles, mobile homes and boats," are ineligible for copyright protection. 37 C.F.R. § 202.11(d). Functionality predominates over aesthetics in the design of these structures.

309. On rare occasions, merger of function and expression may preclude copyright protection for some program code. *See infra* notes 318–319 and accompanying text.

in architectural work cases.[310] To the extent that "design elements [are] attributable to building codes, topography, structures that already exist on the construction site, or engineering necessity[, they] should therefore get no protection."[311] The court observed that methods of construction and good engineering practices were likewise unprotectable by copyright.[312] It further noted that "functional aspects of a work are governed by patent law, not copyright law."[313] Zalewski's infringement claim failed because "[t]here are only so many ways to arrange four bedrooms upstairs and a kitchen, dining room, living room, and study downstairs" for colonial style homes.[314]

Copyright protection in an architectural work extends to the aesthetic design of a building's exterior and interior, but not to utilitarian features such as the layout of electrical, heating, or plumbing infrastructures.[315] There may, ex ante, be many different ways to design the wiring, heating or plumbing systems in a building, but no court would be confused into thinking that a particular layout was protectable expression in the copyrighted building. Those systems lie outside the scope of copyright in any drawing of the building or in the building itself, no matter how much creative effort went into the designs for these systems and no matter how many design choices the architect might have had at the outset. These are functional elements of the architectural design because any creativity in the layout of wires, heating, or plumbing systems is too interconnected with functionality to be part of the expression of copyrighted building.[316]

It would be a doctrinal advance in copyright law if courts articulated a function/expression merger doctrine, as such, in computer program cases.[317]

---

310. Zalewski v. Cicero Builder Dev., Inc., 754 F.3d 95, 105 (2d Cir. 2014) (citing and quoting from *Altai*).

311. *Id*.

312. *Id*. at 105–06.

313. *Id*. at 106 n.19.

314. *Id*. at 107. Note that the court did not invalidate Zalewski's copyright, but regarded the scope of that copyright as sufficiently thin that Cicero did not infringe. *Id.* at 106.

315. *See, e.g.*, COPYRIGHT CLAIMS IN ARCHITECTURAL WORKS, COPYRIGHT OFFICE CIRCULAR 41, at 1–2 (The scope of copyright in copyrighted buildings does not extend to standard configurations of spaces and individual elements such as doors and windows, or "functional elements whose design or placement is dictated by utilitarian concerns.").

316. The existence of alternative ways to achieve the same functionality is routinely disclosed in utility patents which differentiate the claimed invention from the prior art. *See infra* notes 335–37 and accompanying text.

317. Some courts speak of a process/expression distinction in computer program cases. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 836–37 (10th Cir. 1993). This implicitly lays the groundwork for recognition of a process/expression merger doctrine.

The *Altai* decision came close to doing this when the Second Circuit asserted that the merger doctrine precluded copyright protection for efficient design elements of programs.[318] A computer program function/expression merger doctrine could supplement the function/expression merger rule that disqualifies many aesthetic designs of articles of manufacture from copyright protection.[319] It would also complement the fact/expression merger doctrine recognized in recent cases.[320] Merger of ideas and expressions is, as it happens, just one type of merger that courts should (and do) recognize.[321]

Given that programs are utilitarian works that courts often say should enjoy only a thin scope of copyright protection,[322] it is curious that courts have rarely articulated or applied a function/expression merger doctrine, as such, in software infringement cases. Courts' reluctance to do this may stem from the misleading nature of the literary work metaphor that so often pervades copyright discourse about computer programs.[323] This metaphor obscures the deeply functional nature of programs, which have a higher quantum of utilitarian elements as compared with conventional literary works such as novels and plays. While it may sometimes be difficult to distinguish the ideas and the expressions in conventional literary works, there is generally no need to try to separate literary expressions in these works from functionalities because they generally have none.[324]

See also Mitek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1568, 1556 n.19 (11th Cir. 1996); Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839–40 (Fed. Cir. 1992) (recognizing the possibility of process/expression merger).

318. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707–09 (2d Cir. 1992).

319. *See supra* text accompanying notes 298–307. While usually called the separability doctrine, it effectively is a merger doctrine because the opposite of separable is merged.

320. *See, e.g.*, N.Y. Mercantile Exch., Inc. v. Intercontinental Exchange, Inc., 497 F.3d 109, 116–19 (2d Cir. 2007) (NYMEX's creation of settlement prices for futures contracts treated as fact/expression merger); Banxcorp. v. Costco Wholesale Corp., 978 F. Supp. 2d 280, 308 (S.D.N.Y. 2013) (merger doctrine precludes copyright protection of interest rate averages, even though some variation in expression of this fact is possible); *see also* Veeck v. Southern Building Code Congress Int'l, Inc., 293 F.3d 791, 800–02 (5th Cir. 2002) (holding that enacted building codes were unprotectable facts under the merger doctrine). Shubha Ghosh has argued for treating enacted codes as instances of function/expression merger. *See* Shubha Ghosh, *Legal Code and the Need for a Broader Functionality Doctrine in Copyright*, 50 J. COP. SOC'Y 71, 104–08 (2003).

321. *See* Samuelson, *Reconceptualizing Merger*, *supra* note 292, at 438–42.

322. Apple Computer Inc. v. Microsoft Corp., 35 F.3d 1435, 1444 (9th Cir. 1994); *Altai*, 982 F.2d at 712; Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1992).

323. *See, e.g.*, Weinreb, *supra* note 10, at 1167–70.

324. An example of a literary work with separable functionalities is *Red Sparrow* (2013) by Jason Matthews, a spy novel with recipes at the end of each chapter; the book's

Computer programs, by contrast, are virtual machines, "machines . . . that have been constructed in the medium of text."[325] This characterization of programs is not just a metaphor; this is what programs actually are. Characterizing them as "literary works" is not wrong, given the capacious statutory definition of that term,[326] but it is seriously incomplete because it obscures the deeply functional nature of programs and program designs, and ignores the functional behavior that motivates people to purchase them.

Programs are products of an industrial design process.[327] "[C]reating a program is a process of building and assembling functional elements."[328] Unlike physical machines that are constructed from gears, wires, screws, and other hardware components, "programs are built from information structures, such as algorithms and data structures."[329] They are industrial compilations of applied know-how.[330]

Neither copyright nor patent law has conventionally extended protection to these kinds of innovations,[331] although industrial compilations of applied know-how may be and often are protected as trade secrets. Indeed, internal design elements of programs are most often and best protected as secrets, making both copyright and patent protection less needed for nonliteral elements of software. While industrial design elements of programs can and sometimes are reverse-engineered, reverse-engineering of program code does not generally enable the engineer to engage in market-destructive appropriations of program know-how because this method of discovery of program internals is difficult, expensive, and time-consuming.[332]

Command structures and APIs, along with program behavior, are industrial compilations of applied know-how that often cannot be kept secret.[333] Command structures may be visible when users run the program, as in *Borland*, or they may be published, as were the API packages in *Oracle*. But that does not change their essential nature. Functionality and expression are so tightly coupled (i.e., merged) in command structures and APIs that they should lie outside the scope of copyright protection in

---

copyright would not extend to the recipes. *See* Publications Int'l Ltd. v. Meredith Corp., 88 F.3d 473, 480–81 (7th Cir. 1996) (finding recipes in cookbook not entitled to copyright protection).

325. Samuelson et al., *Manifesto*, *supra* note 10, at 2316; *see also id*. at 2316–32 (discussing the nature of programs and their designs).

326. 17 U.S.C. § 101 (definition of "literary works").

327. Samuelson et al., *Manifesto, supra* note 10, at 2329–30.

328. *Id*. at 2321.

329. *Id*.

330. *Id*. at 2326–32.

331. *Id*. at 2342–56.

332. *Id*. at 2389–93.

333. *Id*. at 2356–64.

programs. After all, command structures and APIs are carefully engineered to identify the functions of which programs are capable and the manner in which information must be configured to be exchanged across program boundaries so that programs are able to function properly. As the *Borland* and *Sega* decisions, among others, demonstrate, programmers can generally develop independent implementations of methods, procedures, and processes, such as APIs and command structures, without infringing copyrights. The CAFC failed to recognize this in *Oracle*.

Oracle is among the many plaintiffs in software copyright cases that have characterized computer programs as literary works and insisted that merger is relevant only when the plaintiff had no design alternatives, as the Third Circuit said in *Whelan* and the CAFC in *Oracle*.[334] This approach is unduly restrictive of programmer reuses of functional designs that could contribute to ongoing competition and follow-on innovation in the software industry. Other courts have recognized that there may be more than one method or system that can accomplish a program task, and patents often make reference to prior art that performed the same function in a different way.[335] To be consistent with Supreme Court decisions in other IP cases, the existence of alternative ways to perform a function should not be the sole criterion for whether to treat that type of nonliteral element of a program as expressive, as it was in the CAFC's *Oracle* decision.[336] Also relevant are purpose, cost, and use characteristics of the creation.[337] This is especially true for interfaces necessary for interoperability.

---

334. *See supra* text accompanying notes 28, 65.

335. *See, e.g.*, Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1546, n.29 (11th Cir. 1996) ("The availability of alternatives should not be determinative in distinguishing elements of a computer program that are expression and those that are unprotectable under 102(b). Generally, there is more than one method of operation or process that can be used to perform a particular computer program function . . . . Patents routinely recite prior methods or systems of performing the same function in distinguishing the claimed invention from the prior art."); *see also* Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032, 1042 (9th Cir. 2015) ("[T]he possibility of attaining a particular end through multiple different methods does not render the uncopyrightable [yoga sequences] a proper subject of copyright.").

336. The existence of design alternatives may similarly be a factor in deciding whether a design is too functional to be protected as trade dress, but the Supreme Court has rejected this as a sole criterion for judging the nonfunctionality of trade dress in *TrafFix Devices, Inc. v. Marketing Display Inc. See* 532 U.S. 23, 27–32 (2001). The implications of *TrafFix* for software copyright cases are discussed *infra* text accompanying notes 424–427. *See also* Weinreb, *supra* note 10, at 1170 ("[I]f this rubric [of other possible design choices] is used, copyright effectively absorbs the whole of patent.").

337. *See, e.g.*, *TrafFix*, 532 U.S. at 31–33; Kellogg Co. v. Nat'l Biscuit Co., 305 U.S. 111, 122 (1938) (rejecting claim of trademark in pillow shape of Shredded Wheat biscuits,

C. MERGER MAY BE FOUND WHEN A PLAINTIFF'S DESIGN CHOICES SERVE AS CONSTRAINTS ON THE CHOICES AVAILABLE TO SECOND COMERS

Baker teaches that expression and functionality are sometimes too closely intermixed in some textual works for copyright protection to be available to those elements. When Selden first devised the bookkeeping system embodied in his book, there were other ways to keep books for various accounts, but he was dissatisfied with them. In designing forms for his new system, he wasn't completely free to arrange columns and headings as he wished because state law required all bookkeeping forms to have columns captioned "date," "no.", "to," "for," and "by."[338] Selden's forms complied with this mandate in the five farthest left-hand columns. But he was not constrained in other design decisions for arranging columns and headings in the forms.

Selden believed the selection and arrangement of columns and headings in the new forms was highly creative, as the preface to his book revealed: "To greatly simplify the accounts of extensive establishments doing credit business, and embracing an almost infinite variety of transactions would be a masterly achievement, worthy to be classed among the greatest benefactions of the age."[339] So even though Selden could easily meet a creativity-based originality standard, his forms were nonetheless unprotectable by copyright law because the selection and arrangement of the columns and headings were "necessary incidents" to the system he devised.[340]

This aspect of *Baker* demonstrates that the merger doctrine may apply even when the copyright claimant had other choices when first developing a creative procedure, process, system, or method of operation. Selden's copyright was, of course, valid, but it provided protection only to his explanation of the system, not to the system and the forms that instantiated it. Unlike most useful arts, which are embodied in metal or other materials, Selden's useful art was embodied in a book (or in today's parlance, a literary work). But the Court said in *Baker*: "[*T*]*he principle is the same in all*. The description of the [useful] art in a book, though entitled to the benefit of

despite existence of design alternatives, because "[t]he evidence is persuasive that this form is functional—that the cost of the biscuit would be increased and its high quality lessened if some other form was substituted for the pillow-shape").

338. Samuelson, *Baker v. Selden*, *supra* note 155, at 168.

339. *Id*. at 160 (quoting from the Supreme Court record).

340. Baker v. Selden, 101 U.S. 99, 103 (1879).

copyright, lays no foundation for an exclusive claim to the art itself."[341] The creativity required to develop the system and to devise forms to implement it did not make the system or the forms copyrightable because of the merger of function and expression. By not protecting Selden's system and the forms embodying it, there was breathing room for later bookkeepers to continue to evolve the useful art of bookkeeping and devise new forms that further improved upon Selden's innovation, as Baker himself did.[342]

Courts have followed *Baker*'s admonitions in many subsequent non-software cases.[343] The pertinence of *Baker* for assessing the scope of copyright in software was importantly acknowledged in *Altai*. *Altai* directly invoked *Baker* and the merger doctrine in explaining why efficient functional design elements of programs lie outside of the scope of copyright protection available to programmers.[344] The Second Circuit observed that "[i]n the context of computer program design, the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation."[345] The more efficient a nonliteral element of a program is, the closer it approximates a merger of idea and expression. The court recognized that "hypothetically, there might be a myriad of ways in which the programmer may effectuate certain functions within a program . . . [but] efficiency considerations may so narrow the practical range of choice as to make only one or two forms of expression workable options."[346]

The court in *Altai* was implicitly concerned that the first software developer to devise an efficient functional design for a program should not be able to get 95 years of protection for it and force all other programmers to adopt less efficient solutions. It recognized that "[e]fficiency is an industry-wide goal" for software developers.[347] *Altai* directed that efficient design elements of programs be filtered out of consideration before doing the final step in infringement analysis, even though other design choices might have been available.

Elements of programs that are "dictated by external factors" must also be filtered out.[348] The Second Circuit in *Altai* identified several types of

341. *Id*. at 105 (emphasis added).
342. *See* Samuelson, *Baker v. Selden*, *supra* note 155, at 193; *see also id.* at 169 n.76.
343. *See* Samuelson, *Why Copyright Excludes Systems*, *supra* note 124, at 1936–44 (discussing the cases that followed *Baker*).
344. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707–08 (2d Cir. 1992).
345. *Id*. at 708.
346. *Id*.
347. *Id*.
348. *Id*. at 710.

constraints that might limit the design choices of programmers, including "compatibility requirements of other programs with which a program is designed to operate in conjunction."[349] Although the court referred to the scenes a faire doctrine as a justification for treating compatibility as an external constraint on programmer design decisions,[350] to the extent it used "dictated by" language,[351] the more pertinent doctrine is merger. *Altai* and its progeny regard compatibility as an external factor constraint on design decisions of defendants who are developing programs to interoperate with existing software and hardware.

The CAFC's *Oracle* decision interpreted this external factors constraints category too narrowly. That court considered *Altai* to have directed the filtration of elements dictated by external constraints only insofar as the constraints limited the plaintiff's design choices.[352] Having determined that Sun's engineers were not constrained in designing the Java API packages, the CAFC regarded Google's external constraints argument to be unpersuasive.[353] However, the Second Circuit did not so limit the external constraints category in *Altai*, and in the twenty-three years since

349. *Id*. at 709–10.

350. *See id*. The scenes a faire doctrine is similar to merger in limiting the scope of copyright protection, but it is more focused on whether elements in a protected work are common in works of that kind, not whether they are "dictated" by functional considerations, as merger is in software cases.

351. *See id*.

352. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1370 (Fed. Cir. 2014). The CAFC cited Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc., 307 F.3d 197, 215 (3d Cir. 2002), in support of this interpretation of *Altai*. But in *Grace*, the Court of Appeals for the Third Circuit considered defendant Grace to be a very bad actor who breached license agreements, copied and modified D&B software, and misappropriated its trade secrets. *See Grace*, 307 F.3d 197. In this context, the Third Circuit's rejection of the defendant's claim that its design choices were constrained by what D&B had done should be given little weight.

353. *See Oracle*, 750 F.3d at 1370–71. The CAFC fell back on a *Whelan*-like "is there any other way to do it?" approach to assessing whether the Java API packages were expressive. *Id*. But the CAFC was mistaken on this point. Joshua Bloch, one of the Sun engineers involved in designing the Java API, reports that considerable effort went into developing the Java API to faithfully reimplement the syntax and semantics of Perl so that Perl-trained engineers could more easily work in Java. Sun ran a battery of 30,000 tests to ensure that the Java implementation was consistent with Perl's. Bloch states that the "Java APIs included many preexisting APIs and have since the earliest days of the platform. Many of the original Java APIs were pretty much copied from C to make it easy for C programmers to make the transition." Communication with Joshua Bloch, Sept. 28, 2015 (on file with author).

Altai, numerous cases have treated external constraints as affecting the defendant's as well as the plaintiff's design choices.[354]

Anyone who develops an API for achieving program interoperability is, in effect, creating a constraint on his own subsequent design decisions. At the same time, though, that same API developer is also creating constraints on the design choices of all others who want to develop programs to interoperate with his platform, as the Ninth Circuit implicitly recognized in *Sega*.[355] When Sega initially developed the interface for its Genesis platform, it had many choices about how to construct that interface. But once that interface existed, Sega and its licensees had to conform to it when they made games for the Genesis. Accolade similarly could not make its videogames run on the Genesis platform without reimplementing the Sega interface in its program. That interface constituted the "functional requirements for achieving compatibility."[356] Although the Ninth Circuit characterized "interface procedures" of the Sega program as unprotectable under § 102(b), merger would have been a reasonable alternative ground.[357] Any arguably expressive elements in the Sega interface would be merged with its functionality because third party software cannot execute on the Sega platform unless the interface components exactly conform to the rules that Sega established when designing the interface.[358]

D. SOMETIMES PROGRAM FUNCTIONS MERGE WITH PROGRAM CODE

The interface at issue in *Altai* was a nonliteral element of a program that CA alleged Altai infringed. Sometimes, however, literal copying of code is necessary to achieve interoperability. The Ninth Circuit recognized this in *Sega* because it excused Accolade from infringement for copying a segment of Sega code that was essential to achieving interoperability.[359] The

354. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 709–10 (2d Cir. 1992) (using "dictated by" twice in discussing filtration of external factors). *Sega* and *Borland* are among the decisions in which the defendant's design choices were constrained by the plaintiff's earlier choices. *See also* Samuelson, *Reconceptualizing Merger*, *supra* note 292, at 442–44.

355. *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

356. *See id.* at 1522.

357. *Id*. Note that the "functional requirements for . . . compatibility" phrase uses the language of merger. *See also id*. at 1524 (recognizing that "necessary incidents" components of programs are unprotectable, as are program elements "dictated by the function to be performed").

358. Connectix faced similar constraints when reimplementing the Sony PlayStation interface for its emulation software. *See* Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 602–07 (9th Cir. 2000).

359. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524, 1530–32 (9th Cir. 1992); *see also* Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 995 (N.D. Cal. 2012)

Eleventh Circuit in *Bateman* acknowledged this when ruling that a lower court erred in instructing a jury that only nonliteral elements of a program interface needed for achieving compatibility could be filtered out in applying the AFC test, thus recognizing that sometimes exact copying is truly essential to interoperability.[360] The Sixth Circuit in *Lexmark* likewise struck down a lower court ruling of infringement because exact copying of a short Lexmark program installed in that firm's printer cartridges was a necessary incident to interoperation with the Lexmark printer.[361] The Sixth Circuit expressly relied on merger as the basis for ruling that Lexmark's printer code was ineligible for copyright protection because it was needed to achieve interoperability.[362]

Read together, the *Altai*, *Sega*, *Connectix, Bateman, and Lexmark* decisions not only recognize, but directly hold, that there is a compatibility exception to copyright protection for computer programs when reuse of interface components are necessary for interoperability.[363] This is true whether the interface is embodied in a program or merely specified in a document that is not itself a program. The distinction between APIs and implementations of APIs in independently written code is fundamental in the field of computing.[364] Copyright law protects the code that implements an API, but does not protect the API insofar as it is necessary to enabling

(quoting *Sega*, 977 F.2d at 1524 n.7, as excusing exact copying of a short portion of the Sega code because of its functionality).

360. 79 F.3d 1532, 1546–47 (11th Cir. 1996).

361. Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 542 (6th Cir. 2004) ("[I]f any single byte of the Toner Loading Program is altered, the printer will not function . . . .").

362. *Id*.

363. The court in *Bateman* was unwilling to rule that all APIs were unprotectable as a matter of law, but it directed the lower court to instruct the jury that insofar as program code or APIs were necessary to interoperability, they were not within the scope of protection available to programs. 79 F.3d 1532, 1546–57 (11th Cir. 1996). The interpretation of *Borland* discussed above would also support a compatibility exception. *See supra* Section III.B; *see also* Burk, *supra* note 206, at 591 (suggesting that *Borland* should have been decided on merger of functionality and expression grounds). In *Oracle*, the CAFC distinguished *Sega* and *Connectix* by saying that they were fair use cases and that compatibility considerations might be relevant to fair use defenses. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1368–71 (Fed. Cir. 2014). However, the Ninth Circuit plainly stated in both cases that interface procedures necessary for interoperability are unprotectable elements of copyrighted programs under § 102(b).

364. *See, e.g.*, Alfred Z. Spector, *Software, Interface and Implementation*, 30 JURIM. J. 79, 85–87, 90 (1989) (discussing distinction between interfaces and implementations).

interoperability. Congress has, moreover, indirectly ratified these court holdings that allow reuse of interfaces necessary for compatibility.[365]

Java method headers (or declarations, as they are sometimes called) are not lines of code in the sense of executable program instructions.[366] They are specifications designed to invoke particular program functions. The method headers must be implemented in program instructions that when compiled, will become executable code. In this respect, the method headers are nonliteral elements of programs, which Google reimplemented in independently written code. The functionality of those method headers is inextricably interconnected with any expression they might be said to contain. The rules of Java, as the District Court noted, constrained Google's design choices as to names of Java methods and classes and as to the structure of the API command structure.[367] The merger doctrine was a suitable alternative justification for the holding that the Java APIs at issue in *Oracle* are unprotectable by copyright law, as the District Court held.[368]

E. THE CAFC ERRED IN INTERPRETING THE MERGER DOCTRINE

The CAFC misinterpreted the merger doctrine in several respects. For one thing, it rejected outright the idea that merger can be a "copyrightability" issue.[369] However, merger was in fact treated as a copyrightability issue in *Baker*,[370] as well as in numerous other cases.[371]

---

365. Congress created an exception to rules that forbid bypassing technical protection measures to authorize programmers to reverse engineer technically protected programs to get access to interface information necessary for interoperability. 17 U.S.C. § 1201(f).

366. *See, e.g.*, Mike Masnick, *Yes, The Appeals Court Basically Got Everything Wrong in Deciding APIs Are Covered by Copyright*, TECHDIRT (Aug. 18, 2015), https://www.techdirt.com/articles/20150817/11362131983/yes-appeals-court-got-basically-everything-wrong-deciding-apis-are-covered-copyright.shtml [https://perma.cc/9VY6-RN9V] (criticizing the CAFC *Oracle* decision for not understanding what an API is).

367. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 979 (N.D. Cal. 2012) ("[S]ince there is only one way to declare a given method functionality, everyone using that function must write that specific line of code in the same way . . . .").

368. *Id.* at 997.

369. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1354–62 (Fed. Cir. 2014).

370. Baker v. Selden, 101 U.S. 99, 103 (1879) (Selden's forms held unprotectable by copyright law as "necessary incidents" to the bookkeeping system).

371. *See, e.g.*, Ho v. Taflove, 648 F.3d 489, 497 (7th Cir. 2011) (equation not copyrightable); ATC Distrib. Group, Inc. v. Whatever It Takes Transmission & Parts, Inc., 402 F.3d 700, 707–08 (6th Cir. 2005) (parts numbering system not copyrightable); Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 534–42 (6th Cir. 2004) (printer program not copyrightable); Warren Publ'g Inc. v. Microdos Data Corp., 115 F.3d 1509, 1518 n.27 (11th Cir. 1997) (systematic compilation not copyrightable); Kern River Gas Co. v. Coastal Corp., 899 F.2d 1458, 1463–65 (5th Cir. 1990) (route of gas line not copyrightable). The Copyright Office also recognizes that merger can present a

Second, the CAFC erred in saying that merger can only be found when, ex ante, there is just one way to express an idea or function and any creativity in the design of an API makes it protectable by copyright law.[372] Other courts, *Altai* among them, have recognized that there sometimes is, ex ante, more than one or a few ways to design nonliteral elements of computer programs, but efficiency considerations may narrow the range of options, so that merger applies.[373] The CAFC did not consider whether efficiency considerations might have limited options as to the design of the Java API method headers. Numerous cases have taken other factors into account when assessing merger defenses, such as whether the claimed expression was the most logical and useful way to do something, whether practical considerations or functionality limited options, and whether particular designs were necessary to achieving objectives.[374]

The CAFC in *Oracle* conjectured that merger might apply to three "core" Java API packages that were necessary to make use of the Java language, which all agreed Google was free to use.[375] The design of those core packages was probably no more constrained, ex ante, than any others of the API packages. Yet, if the Java language cannot be used at all unless

---

copyrightability issue. *See* Compendium, *supra* note 143, § 313.3(B) (stating Office will not register claims of copyright when it believes the work consists of merged expression). The CAFC relied upon Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068 (9th Cir. 2000), to say that the Ninth Circuit does not recognize merger as a copyrightability issue. *See Oracle*, 750 F.3d at 1358. While *Ets-Hokin* did reject the merger defense that the photograph at issue was uncopyrightable, this was because the photograph had enough originality to support a copyright. *See* 225 F.3d at 1077, 1082. The scope of that copyright, however, was so "thin" that another photograph of the bottle did not infringe. *See* Ets-Hokin v. Skyy Spirits, Inc., 323 F.3d 763, 766 (9th Cir. 2003). In *Allen v. Academic Games League of Am.*, the Ninth Circuit treated merger as a copyrightability issue. *See* 89 F.3d 614, 617–18 (9th Cir. 1996); *see also* Samuelson, *Reconceptualizing Merger*, *supra* note 292, at 435–38.

372. *Oracle*, 750 F.3d at 1359–62.

373. *See supra* text accompanying notes 103–106. *ATC* is an example of a non-software copyright case recognizing efficiency as a design constraint. *See ATC*, 402 F.3d at 709.

374. *See, e.g.*, *id.* at 707 (reasonableness as constraint); Rice v. Fox Broad. Co., 330 F.3d 1170, 1177 (9th Cir. 2003) (merger if most logical); Yankee Candle Co. v. Bridgewater Candle Co., 259 F.3d 25, 35 (1st Cir. 2001) (functional considerations); Matthew Bender & Co. v. West Publ'g, 158 F.3d 674, 685 (2d Cir. 1998) (feature became standard); Crume v. Pac. Mut. Life Ins. Co., 140 F.2d 182, 184 (7th Cir. 1944) (alternative language might not achieve objective); Matthew Bender & Co. v. Kluwer Law Book Publishers, Inc., 672 F. Supp. 107, 110–11 (S.D.N.Y. 1987) (most logical; practical considerations).

375. *Oracle*, 750 F.3d at 1362. Because of interdependencies among elements of the Java API packages, reuse of at least ten is necessary to implement the Java language specification.

one conforms to the method headers and classes of those three packages, merger would be a reasonable doctrine to apply.

Third, the CAFC incorrectly stated that merger can only be found when the plaintiff (not the defendant) faced constraints in its design decisions.[376] But courts often recognize that a defendant's design choices can be constrained by what the plaintiff did.[377] To compete effectively and to enable ongoing innovation, a second comer may need to use the same designs. Even CONTU accepted this proposition: "when specific instructions, *even though previously copyrighted*, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement."[378] This indicates that CONTU accepted that function and expression might merge over time.

Fourth, the CAFC ignored the District Court's finding that there was far less of a distinction between the Java language and the API packages than Oracle acknowledged.[379] Insofar as there is little or no difference between the Java language—which all agree is not protectable by copyright law—and the component parts of the API that Google used, this consideration weighs in favor of finding merger in *Oracle*.[380]

The CAFC should also have taken into account the interests of the nine million Java programmers who have become accustomed to using the Java command structure when expressing themselves in the Java language.[381] If

376. *Id*. at 1361.

377. *See, e.g.*, N.Y. Mercantile Exch., Inc. v. Intercontinental Exchange, Inc., 497 F.3d 109, 116–19 (2d Cir. 2007); *ATC*, 402 F.3d at 705–09; Southco, Inc. v. Kanebridge Corp., 390 F.3d 276, 282 (3d Cir. 2004); Lexmark Int'l, Inc. v. Static Control Components, 387 F.3d 522, 536–42 (6th Cir. 2004); s*ee also* Samuelson, *Reconceptualizing Merger*, *supra* note 292, at 442–44.

378. CONTU Report, *supra* note 2, at 74 (emphasis added).

379. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 982 (N.D. Cal. 2012).

380. The CAFC also failed to recognize that numerous cases have treated merger as reasonably interchangeable alternatives to § 102(b) defenses. *See, e.g.*, Ho v. Taflove, 648 F.3d 489, 497 (7th Cir. 2011); Warren Publ'g Inc. v. Microdos Data Corp., 115 F.3d 1509, 1518 n.27 (11th Cir. 1997); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 845 (10th Cir. 1993). As in *Baker*, once an author chooses to create a text embodying an unprotectable system or method of operation, the expressive choices of follow-on creators wanting to use the same system or method are constrained.

381. There is a deep irony in Oracle's copyright lawsuit against Google. Sun Microsystems, whose IP assets Oracle acquired in 2010, was once the foremost proponent of freedom to interoperate, by which it meant there should be no intellectual property protection for APIs insofar as they were necessary to enable interoperability. Sun's Deputy General Counsel Peter M.C. Choy was a lead lawyer on numerous amicus curiae briefs for the American Committee for Interoperable Systems in software copyright cases, including *Altai*, *Sega, Bateman*, and *Borland*. These briefs can be found online. *See Interoperability Resources*, COMPUTER & COMMUNICATIONS INDUSTRY ASSOCIATION, http://www.ccia

it was appropriate to take into account the interests of users who constructed macros in the Lotus 1-2-3 language in denying Lotus's claim in *Borland*, then it should be appropriate to take into account the third party effects of a ruling in Oracle's favor on programmers accustomed to using the Java command structure that Google decided to include in the Android software and educators who teach them to students. Courts should not be forcing programmers to engage in needless variation when standardization would better accomplish the objectives of copyright law by letting programmers express themselves in the command language they know well.[382]

---

net.org/interop/ [https://perma.cc/L5QK-BDXP] (compiling amicus briefs on the interoperability issue).

Consider this excerpt from the ACIS brief to the Supreme Court in support of Borland, which echoes arguments that Google made in *Oracle*:

> Unlike traditional literary works such as novels and plays that stand alone and do not need to interact with any other work, computer programs never function alone; they function only by interacting with the computer environment in which their developers place them. This environment is absolutely unforgiving. Unless the computer program conforms to the precise rules for interacting with the other elements of the system, no interaction between the program and the system is possible. As a consequence, no matter how much better or cheaper the new program is, it will not enjoy a single sale if it cannot interoperate in its intended environment. If the developer of one part of the environment can use copyright law to prevent other developers from writing programs that conform to the system of rules governing interaction within the environment – interface specifications, in computer parlance – the first developer could gain a patent-like monopoly over the system without ever subjecting it to the rigorous scrutiny of a patent examination. Lotus seeks to use copyright in exactly this manner.

Brief Amici Curiae of American Committee for Interoperable Systems and Computer & Communications Industry Ass'n in Support of Respondent at 4–5, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728487. Oracle cannot perhaps be bound by the legal positions Sun took in those cases, but surely it is fair game to point out the stark contrast between then and now. After all, Google talked to Sun about a license, not Oracle. Moreover, Sun's last CEO testified in support of Google's defense. *See* Bryan Bishop, *Former Sun CEO Jonathan Schwartz Testifies for Google in Oracle Trial*, THE VERGE (Apr. 26, 2012), http://www.theverge.com/2012/4/26/2977858/former-sun-ceo-jonathan-schwartz-testifies-for-google-oracle-trial [https://perma.cc/8F9R-YH25].

382. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 818 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996) (stating it would be "absurd" to require users to have to learn new command structures for programs); Burk, *supra* note 206, at 592 (In *Baker* and *Borland*, protecting the forms and commands "would have been tantamount to protecting the method or process they embodied. This is the rationale of merger."); see also *supra* text accompanying notes 284–286 concerning the difficulties that Java programmers would have encountered if Google had attempted to develop different method headers and classes for a variant version of them for the Android platform.

Command structures and APIs are examples of computer program design elements as to which merger of function and expression may and often does occur. It is important to competition and ongoing innovation in the software industry that merged elements such as these are available for reuse by subsequent programmers, especially given the powerful presence of network effects.[383]

V. DIFFERENT CONCEPTUALIZATIONS ON THE RELATIONSHIP BETWEEN PATENT AND COPYRIGHT PROTECTIONS FOR SOFTWARE

Courts have sometimes assigned to copyright law the role of protecting program expression and patent law the role of protecting program processes or other functionality.[384] The Ninth Circuit's *Sega* decision, for instance, stated that Sega could not use copyright law to get exclusive rights in the interfaces that constituted the functional requirements for achieving program-to-program compatibility because that kind of protection was available only from patent law.[385] The Second Circuit in *Altai* suggested that patents might be more appropriate than copyright for protecting utilitarian nonliteral elements of programs.[386] Such statements draw upon the Supreme Court's *Baker* decision, which channeled useful arts to the

383. A prominent economist has noted that it is "inefficient to protect the arbitrary choices whose commercial value is created solely by the network incentives to imitate—and to protect the useful ideas only indirectly by protecting these ancillary innovations. Such protection not only seems likely to have adverse consequences on compatibility, but also protects only indirectly and haphazardly the useful ideas, the costs of whose creation intellectual-property policy is meant to cover." Joseph Farrell, *Standardization and Intellectual Property*, 30 JURIM. J. 35, 49 (1989).

384. *See, e.g.*, Mitek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1556 n.19 (11th Cir. 1996) (copyright protects expression, but not program processes which are the province of patent law); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 837 (10th Cir. 1993) (description of program process may be copyrightable, but program process is patentable); Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839 (Fed. Cir. 1992) (copyright protects program expression, and patent law protects program processes and methods). Some commentators have expressed this view as well. *See, e.g.*, Dennis S. Karjala, *The Relative Roles of Patent and Copyright in the Protection of Computer Programs*, 17 J. MARSHALL J. COMP. & INFO. L. 41, 41–42 (1998); Englund, *supra* note 10, at 893–96.

385. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992); *see also* Apple Computer Inc. v. Microsoft Corp., 35 F.3d 1435, 1443 (9th Cir. 1994) ("Apple cannot get patent-like protection for the idea of a graphical user interface . . . .").

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

patent system and limited the scope of copyright in any work describing or depicting the useful arts to the plaintiff's expression.[387]

The patentability of a process, system or method of operation embodied or depicted in a copyrighted work should, in keeping with the Supreme Court's ruling in *Baker*, be a factor indicating that that innovation is among the functional design elements that § 102(b) was meant to exclude from copyright protection. It would undermine incentives to use the patent system if innovators could get several times longer duration of IP protection from copyright law without applying for a patent and subjecting their claims to examination for novelty and nonobviousness, among other things.[388] Making sure that copyright does not indirectly protect technological innovations that are not, in fact, patented promotes competition and ongoing innovation in the useful arts.

Unfortunately, the functional and expressive elements of computer programs cannot be as readily distinguished as *Baker* and the conventional paradigms of copyright and patent law posit.[389] Functionality pervades program design, and creative choices abound, so the usual channeling mechanisms that courts have traditionally applied work less well as applied to computer programs. Courts should develop more subtle ways to conceptualize the relative roles of utility patent and copyright in protecting program innovations than the categorical exclusivity approach that *Baker* and its progeny law suggest.[390]

Although *Baker* posited that patent and copyright laws protect very different kinds of innovations, Section A explains why courts have not always found categorical exclusivity of intellectual property subject matters to be persuasive and why separating out the roles of patent and copyright in the protection of software innovations has proven to be difficult. Section B

387. Baker v. Selden, 101 U.S. 99, 101–03 (1879).

388. *See, e.g.*, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 984, 996 (N.D. Cal. 2012); Karjala, *supra* note 384, at 44–45.

389. *See, e.g.*, Reichman, *supra* note 1, at 2480–81.

390. Inventors of new and useful machines, manufactures, compositions of matter, and processes are eligible to apply for utility patent protection. 35 U.S.C. § 101, et seq. If the patent applicant satisfies the statutory requirements, including articulation of specific claims that will define the scope of the patent, a utility patent will issue. Utility patents have a maximum duration of twenty years from the date of filing. 35 U.S.C. § 154(a)(2). The patentability of computer program innovations has been controversial for decades. *See, e.g.*, Pamela Samuelson, Benson *Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025 (1990). The Supreme Court recently struck down software-related patent claims in Alice Corp. v. CLS Bank, 573 U.S. ___, 134 S. Ct. 2347 (2014). Patent protection may also be available to creators of ornamental designs for articles of manufacture. 17 U.S.C. § 171, et seq.

explains the flaws in the CAFC's *Oracle* decision's analysis of copyright-patent boundary issues. Section C offers an alternative framework for thinking about those boundaries as applied to software innovations. Section D observes that software developers often rely more on non-IP strategies, such as first mover advantages and complementary assets, to attain competitive advantage than on intellectual property protections.

A. REASONS TO BE CAUTIOUS OF CATEGORICAL EXCLUSIVITY ARGUMENTS ABOUT PATENT AND COPYRIGHT PROTECTIONS FOR SOFTWARE INNOVATIONS

There are several reasons why it may be risky for defendants in software copyright cases to argue that a particular nonliteral element of the plaintiff's program (which it copied) cannot qualify for copyright protection because patents have issued for that kind of program innovation.[391] For one thing, *Baker* notwithstanding, defendant arguments for categorical exclusivity in intellectual property cases have sometimes proven unpersuasive.[392] For instance, Mazer lost his argument that Stein's Balinese dancer statuette was ineligible for copyright protection because Stein could have gotten (but did not) a design patent on the statuette for use as a lamp base.[393] The Court was untroubled by the existence of this overlap between design patent and copyright subject matters.[394] Stein's statuette was an ornamental design for an article of manufacture, but it also qualified as a copyrightable work of art.[395] This explains the Court's dictum that the potential patentability of

391. Google's brief asking the Supreme Court to review the CAFC ruling seemed to make a categorical exclusivity argument. *See* Petition for a Writ of Certiorari at 23–32, Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5319724.

392. *See* J.E.M. Ag Supply, Inc. v. Pioneer Hi-Bred Int'l, Inc., 534 U.S. 124 (2001) (rejecting argument that novel plants could not be patented because Congress intended for them to be protected only under the Plant Variety Protection Act).

393. Mazer v. Stein, 347 U.S. 201, 217 (1954).

394. *See id*. at 214–15. The Court cited to precedents recognizing an overlap of design patent and copyright subject matter. *Id*. at 215 n.33, 217 n.37. Some of these cases had required creators to elect between copyright and design patent protection; the Court chose not to address the election issue. *See id*.

395. There is not much overlap of copyright and design patent subject matters because of copyright's useful article doctrine which excludes PGS works in which expression and functionality have merged. *See supra* notes 297–304 and accompanying text. The integration of functionality and ornamentality does not disqualify designs from design patent protection. *See, e.g*., Robert C. Denicola, *Applied Art & Industrial Design: A Suggested Approach to Copyright in Useful Articles*, 67 MINN. L. REV. 707, 707–08 (1982) ("Design patents long offered the possibility of protection for the ornamental design of a useful product.").

that design was irrelevant to whether it was eligible for copyright protection.[396]

However, neither the Supreme Court nor any other court has recognized subject matter overlap between copyright and utility patent laws.[397] No decision has ever held that a creator can get copyright *and* utility patent protection for *exactly* the same aspect of a creation.[398] The *Baker* decision suggests that subject matter overlap of these two laws does not exist. Yet, when the Supreme Court was presented with a categorical exclusivity argument in *Borland*, it split evenly on the merits despite Borland's citation to some utility patents on similar innovations as evidence that they were patent, not copyright, subject matter.[399]

A second reason not to put too much weight on the existence (or not) of utility patent protection for some types of program-related innovations in judging whether copyrights have been infringed is that patents on some innovative designs may have issued at a different level of abstraction than the copyright claim may be alleged to cover.[400] The District Court in *Oracle* did not, for example, analyze the API patents it mentioned to compare them to the API command structures in which Oracle claimed copyright.[401]

Yet, a levels-of-abstraction assessment of the relative roles of patent and copyright in protecting programs may not be a reliable indicator. After all, patent lawyers make strategic decisions in drafting patent applications about the level of abstraction at which to pitch their clients' claims. There are significant advantages to claiming inventions at higher levels of abstraction because if the claim is allowed, it will enable the patentee to enjoy a broader scope of patent protection for the innovation.[402] Even if patent claims could

396. *Mazer*, 347 U.S. at 216–17.

397. *Mazer* distinguished design patents from utility patents in relation to copyright protections. *Id*. at 215 n.33, 217 (citing approvingly to *Taylor Instrument* for its holding that utility patents and copyrights are mutually exclusive).

398. *See, e.g.*, Laureyssens v. Idea Group, Inc., 964 F.2d 131, 141 (2d Cir. 1992) (patents on puzzle design narrowed scope of copyright in plaintiff's puzzle).

399. *See, e.g.*, Brief for Respondent, *supra* note 200, at 21, 32–34, 43–44. The categorical exclusivity of patent and copyright subject matters has a constitutional character because the Constitution speaks of Congress as having power to grant exclusive rights to authors and to inventors in "their *respective* writings and discoveries." U.S. CONST., art. I, § 8, cl. 8 (emphasis added).

400. *See, e.g.*, Lemley, *supra* note 17, at 22–27 (discussing the role of software patents and copyrights and levels of abstraction as a way to distinguish their roles in software protection).

401. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 996 (N.D. Cal. 2012).

402. *See, e.g.*, Mark A. Lemley, *Software Patents and the Return of Functional Claiming*, 2013 WISC. L. REV. 905 (2013).

have been drafted to cover lower level functionality, they need not be so limited.

A third reason to be cautious about the patent or copyright subject matter issue in computer program cases is that § 102(b) excludes from the scope of copyright protection more than just patentable procedures, processes, systems, and methods of operation.[403] Mathematical innovations are among the fundamental building blocks of knowledge that should be free for reuse as abstract ideas under both copyright and patent law.[404] Consider, for instance, Benson's algorithms for transforming binary coded decimals to pure binary form, which the Court held was unpatentable as an abstract idea.[405] That algorithm would unquestionably be part of the SSO of any program that embodied it. If the Benson algorithm is too abstract to qualify for patent protection, it should be a strong candidate for ineligibility for copyright protection under § 102(b) as an abstract mathematical procedure.

A fourth consideration that cuts against categorical exclusivity is that software-related patents might have issued in error.[406] This is especially pertinent now that the pendulum on the patentability of software has gone from almost-never-available in the 1960s to mid-1980s to almost-always-available in the mid-1980s to the early 2000s.[407] In the past decade, that

403. *See, e.g.*, Ho v. Taflove, 648 F.3d 489, 497 (7th Cir. 2011) (equation held unprotectable by copyright law). Parts numbering systems are similarly ineligible for copyright protection and likely unpatentable as well. *See, e.g.,* Southco, Inc. v. Kanebridge Corp., 390 F.3d 276, 281 (3d Cir. 2004) (parts numbering system unprotectable).

404. Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 573 U.S. ___, 134 S. Ct. 2347, 2355 (2014) (patent); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–43 (10th Cir. 1993) (copyright).

405. Gottschalk v. Benson, 409 U.S. 63, 67, 71–72 (1972). The Supreme Court has recently reaffirmed that the Benson algorithm was unpatentable subject matter. *See Alice*, 134 S. Ct. at 2354–55 (citing approvingly to *Benson*).

406. *See, e.g.*, Guthrie v. Curlett, 10 F.2d 725 (2d Cir. 1926) (invalidating patent on consolidated tariff index held on novelty grounds). After losing the patent case, Guthrie sued Curlett for copyright infringement. Although Guthrie's copyright was valid, the court held that Curlett only copied non-copyrightable functional elements and therefore had not infringed. *See* Guthrie v. Curlett, 36 F.2d 694 (2d Cir. 1929). The invalid patent was not mentioned in the copyright opinion. The patents to which Borland pointed in its brief to the Supreme Court may also have issued in error. *See* Brief for Respondent, Lotus Dev. Corp. v. Borland Int'l Inc., 516 U.S. 233 (1996) (No. 94-2003), 1995 WL 728538; *see also supra* note 200 and accompanying text.

407. For a discussion of the pre-1990 software patent case law, see Samuelson, Benson *Revisited*, *supra* note 390, at 1048–1132. For a discussion of the case law on patent subject matter in the 1990s and 2000s, see Michael Risch, *Everything is Patentable*, 75 TENN. L. REV. 591 (2008).

pendulum has swung back to sometimes-available-but-sometimes-not.[408] During the mid-1980s to the mid-2000s, during the almost-always-available period, the USPTO issued a large number of patents for software innovations.[409] Quite a few of these have been struck down in the aftermath of the Supreme Court's decision in *Alice Corp. v. CLS Bank Int'l*, which affirmed the invalidation of patents on software-implemented method and system claims for managing settlement risks for financial transactions.[410]

B. THE *ORACLE* DECISION'S ANALYSIS OF COPYRIGHT-PATENT BOUNDARIES WAS FLAWED

Utility patent and copyright laws are, of course, separate laws, and each has a different role to play in protecting intellectual creations, including computer programs. Even if the possibility of patents on certain types of software innovations should not automatically mean that those innovations cannot be copyrighted, competition and innovation policies, as well as freedom of expression policy, should caution against collapsing legal boundaries so that there is substantial or complete overlap in copyright and utility patent subject matters.[411]

Yet, the CAFC seems to have done just that in response to Google's API-as-patent-not-copyright-subject-matter argument.[412] It invoked the *Mazer* dictum that "[n]either the Copyright Statute nor any other says that because a thing is patentable it may not be copyrighted,"[413] seemingly untroubled by the idea of overlapping utility patent and copyright protections for API designs. Had the CAFC read *Mazer* more carefully, however, it would have noticed that in the very next sentence, the Court reaffirmed that utility patents and copyrights were quite different because copyright law cannot be used to protect patentable ideas, only authorial

408. *See, e.g.*, DDR Holdings LLC v. Hotels.com L.P., 773 F.3d 1245, 1259 (Fed. Cir. 2014) (upholding software patent and finding no § 101 subject matter problems). See *infra* note 410 for examples of software patents that have been struck down since *Alice*.

409. *See, e.g.*, Brief Amicus Curiae of IEEE-USA in Support of Grant of Certiorari at 2, Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 573 U.S. ___, 134 S. Ct. 2347 (2014) (No. 13-298), 2013 WL 5555082 (nearly one million software patents have issued).

410. *See* 134 S. Ct. 2347, 2360 (2014). For patents that have been struck down since *Alice*, see, for example, Content Extraction & Transmission LLC v. Wells Fargo Bank, 776 F.3d 1343 (Fed. Cir. 2014) (data storage method); Ultramercial, Inc. v. Hulu LLC, 772 F.3d 709 (Fed. Cir. 2014) (Internet advertising method); buySAFE, Inc. v. Google Inc., 765 F.3d 1350 (Fed. Cir. 2014) (transaction guaranty method).

411. *See, e.g.*, Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 27 (2001) ("As patent and copyright law overlap more and more, it becomes critical that they take account of each other.").

412. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1379–81 (Fed. Cir. 2014).

413. *Id.* at 1380 (quoting Mazer v. Stein, 347 U.S. 201, 217 (1954)).

expression.[414] *Mazer* also cited approvingly to *Baker*-inspired precedents holding that, as the Court put it, "the Mechanical Patent Law and Copyright Laws are mutually exclusive."[415]

The CAFC discussed the patent-not-copyright-subject-matter issue at the very end of its *Oracle* decision, characterizing it as one of Google's "policy-based arguments."[416] These arguments, the CAFC said, "appear premised on the belief that copyright is not the correct legal ground upon which to protect intellectual property rights in software programs," as though "patent protection for such programs, with its insistence on nonobviousness and shorter terms of protection, might be more applicable and sufficient."[417] But that was not what Google was arguing. Its two-fold point was that patents were a more appropriate way than copyright to protect APIs and that copyright could protect the code that implemented an API, but not the API.[418]

To support its view that copyright was more suitable for protecting programs than patents, the CAFC cited two journalistic articles.[419] With a rhetorical flourish, the CAFC went on to say that until the Supreme Court or Congress decided to alter IP rules, it felt bound to enforce copyright protection for programs and "decline[d] any invitation to declare that protection of software programs should be the domain of patent law, and only patent law."[420] The CAFC should instead have recognized that "the existence of software patents should make courts less willing to extend the

414. *Mazer*, 347 U.S. at 217 ("Unlike a patent, a copyright gives no exclusive right to the art disclosed.").

415. *Id*. at 215 n.33 (citing Taylor Instrument Cos. v. Fawley-Brost Co., 139 F.2d 98 (7th Cir. 1943) and Brown Instrument Co. v. Warner, 161 F.2d 910 (D.C. Cir. 1947)). The Court also cited two others of *Baker*'s progeny, Fulmer v. United States, 103 F. Supp. 1021 (Ct. Cl. 1952) (making parachute did not infringe on copyright in parachute design) and Muller v. Triborough Bridge Authority, 43 F. Supp. 298 (S.D.N.Y. 1942) (construction of a bridge did not infringe on copyright in bridge design plan). *Mazer*, 347 U.S. at 217 n.39. These decisions are also more consistent with exclusivity of copyright and patent subject matter than to overlap.

416. *Oracle*, 750 F.3d at 1379–81.

417. *Id*. at 1379–80.

418. Masnick, *supra* note 366.

419. *Oracle*, 750 F.3d at 1380. The CAFC cited to one article published in the Economist magazine and another in the Washington Post in support of copyright as the better form of protection for software. *Id*. While the CAFC correctly cited to my *CONTU Revisited* article, *supra* note 2, as recommending a sui generis form of protection for programs instead of copyright, it did not cite the most relevant of my articles in which I affirm that copyright protects program code, but should not protect interfaces necessary for interoperability. *See* Samuelson, *Why Copyright Excludes Systems*, *supra* note 124, at 1962–74.

420. *Id*. at 1381.

coverage of copyright law to ideas and functional elements of programs and more willing to engage in a strict filtration analysis,"[421] especially in cases involving claims of nonliteral infringements.

The CAFC's *Oracle* opinion may reflect that court's anxiety that software would be underprotected by IP law if it ruled in Google's favor so soon after the Supreme Court's *Alice* decision substantially cut back on the availability of patent protection for software-related inventions.[422] But the CAFC's *Oracle* decision is at odds not only with *Baker* and a fair reading of *Mazer*, but also with two of the CAFC's prior decisions in software copyright cases as well as other software copyright decisions.[423]

C. AN ALTERNATIVE APPROACH TO CONCEPTUALIZING THE ROLES OF COPYRIGHTS AND PATENTS IN PROTECTING SOFTWARE INNOVATIONS

A more appropriate way to conceptualize the respective roles of utility patent and copyright protection for computer programs may be one akin to the approach the Supreme Court took when presented with an argument about a potential overlap between patent and trademark protection in *TrafFix Devices, Inc. v. Marketing Displays, Inc.*[424] TrafFix argued that the existence of an expired patent on a dual spring design to enable roadside signs to withstand and bounce back from big gusts of wind meant that the patented design was ineligible to be protectable as trade dress.[425] The Court considered the expired patent to be "strong evidence" that the spring design was too functional to be eligible for trademark protection.[426] However, it did not go so far as to rule that the existence of a utility patent for a particular design automatically disqualified it from trade dress protection.[427]

---

421. Lemley, *supra* note 17, at 27; *see also* Karjala, *supra* note 384, at 66–69 (arguing that patent protection is more appropriate than copyright for computer program SSO because it is more functional, rather than aesthetic, in nature).

422. Judge O'Malley, who wrote the *Oracle* decision, would have upheld as patentable subject matter all of the claims that Alice made against CLS Bank. *See* CLS Bank Int'l v. Alice Corp., 717 F.3d 1269, 1292–1321 (Fed. Cir. 2010). She joined three of the five pro-patent opinions in *Alice*. Her concerns notwithstanding, the Supreme Court struck down all of Alice's claims. Neither Judge Plager nor Taranto participated in the CAFC's *Alice* decision.

423. *See, e.g.*, *Atari Games*, 975 F.2d at 838–39; *see also Hutchins*, 492 F.3d at 1383–85. See *supra* note 384 for citations to other software copyright decisions distinguishing copyright and patent protection for software.

424. TrafFix Devices, Inc. v. Marketing Displays, Inc., 532 U.S. 23 (2001).

425. *Id*. at 27–28.

426. *Id*. at 29–30. The Court recognized that the very same aspect of the design that MDI claimed as trade dress fell within the scope of the expired patent; it also considered the description of the functional advantages of the design in the patent. *Id*. at 31–32.

427. *Id*. at 35.

Along similar lines, courts in software copyright cases, when presented with evidence that utility patents have issued for the same type of nonliteral element of program design as the plaintiff argues is protectable expression, should consider those patents as relevant evidence about whether the innovation in question is a method or system excluded from copyright protection under § 102(b).[428]

Consistent with *TrafFix* and numerous software copyright decisions, the District Court pointed to the patents Oracle and its predecessor Sun Microsystems had obtained on some aspects of the Java API as a reason not to extend copyright protection to them,[429] asserting that "this trial showcases the distinction between copyright protection and patent protection" for computer program innovations.[430] The issue "loom[ed] large, where, as here, the vast majority of the code was not copied and the copyright owner must resort to alleging that the accused stole the 'structure, sequence, and organization' of the work."[431] The court later noted that software copyright cases decided in recent years had moved away from using "SSO" as a characterization of protectable elements of programs out of "fidelity to Section 102(b) and recognition of the danger of conferring a monopoly by copyright over what Congress expressly warned should be conferred only by patent."[432]

The District Court recognized that copyright owners might try to claim exclusive rights to "a functional system, process or method of operation that belongs in the realm of patents, not copyrights."[433] This troubled the District Court because patent protection, when available, was of much shorter duration than copyright, and unlike copyrights which provide automatic protection, patents are only available to those who apply and have their claims examined for novelty and nonobviousness, more stringent standards of eligibility than copyright requires.[434] To buttress its opinion on this point, the District Court quoted from *Baker* and *Sega* about the dangers of allowing creators to get patent-like monopolies through copyright

428. *See, e.g.*, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 997–98 (N.D. Cal. 2012). Courts might also usefully consider whether the possible ineligibility of program SSO for patenting under *Alice* and other precedents on account of abstractness should mean that this nonliteral element of a program is also ineligible for copyright protection under § 102(b).

429. *See id.* at 996.

430. *Id.* at 984.

431. *Id*.

432. *Id*. at 996.

433. *Id*. at 984.

434. *See id*.

protection.[435] It noted that large numbers of patents had issued in recent years for software innovations, and indeed, both Oracle and Sun had gotten patents on some aspects of the Java API (although Oracle did not claim that Google infringed any of them).[436]

The District Court noted that Oracle made much of the creativity that went into the design of the Java APIs, but this was beside the point.

> Inventing a new method to deliver a new output can be creative, even inventive, including the choices of inputs needed and outputs returned . . . . But such inventions—at the concept and functionality level—are protectable only under the Patent Act . . . . Based on a single implementation, Oracle would bypass this entire patent scheme and claim ownership over any and all ways to carry out methods for 95 years . . . .[437]

The District Court did not consider the patent-not-copyright-subject-matter issue as an independent ground for its ruling. Rather, the consideration merely reinforced the court's conclusion that the command structure at issue was a system or method of operation excluded from copyright protection under § 102(b). Closer scrutiny of interface patents and a comparison of them with the SSO at issue in *Oracle* might have made that court's analysis more persuasive. Nonetheless the District Court's approach to the patent-not-copyright-subject-matter issue was much sounder than the CAFC's.

D. SOFTWARE DEVELOPERS ATTAIN COMPETITIVE ADVANTAGE BEYOND IP RIGHTS

The CAFC should have been less anxious than they seemingly were in *Oracle* about the receding role of patents and the necessarily thin scope of copyright protection for program innovations because software developers have a multi-faceted, nuanced approach to attaining competitive advantage in the marketplace. A recent empirical study demonstrates that software entrepreneurs consider first mover advantages the most important way to attain advantage.[438] Complementary assets (e.g., providing services or customization) are next most important.[439] Entrepreneurs rated copyrights, trademarks, and trade secrecy as equally significant in protecting software,

435. *See id*. at 984, 994–96.
436. *Id*. at 996.
437. *Id*. at 998.
438. Stuart J.H. Graham et al., *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L.J. 1255, 1290 (2009).
439. *Id*.

but only between slightly and moderately important to gaining competitive advantage.[440] Only a minority of software entrepreneurs owned or were seeking patents.[441] Software patents were rated just over slightly important.[442] Patents were mainly valued as useful assets for impressing investors.[443] It is also important to recognize that software developers are often able to recoup investments through business models that depend very little on intellectual property rights,[444] such as Google's ad-revenue strategy for recouping its investment in the Android platform.[445] IP rights play a more modest role in protecting software innovations than many IP lawyers might expect.

VI. REFINING THE TESTS FOR SOFTWARE COPYRIGHT INFRINGEMENT

This Article has analyzed various doctrinal approaches that courts have taken in adjudicating infringement claims in software copyright cases. Courts have generally sought to interpret copyright rules in keeping with traditional principles of that law, but also with an eye to providing sufficient protection for programs to induce investments in their development while leaving breathing room for subsequent programmers to build on what has come before in developing competing or complementary innovations. The outcomes of the software copyright decisions have generally been consistent, even though the doctrinal hooks courts have employed have differed in some respects.[446]

Many disputes have been resolved by applying *Altai*'s AFC test, while other cases have relied on the § 102(b) method and process exclusions, the merger doctrine, or fair use. Regardless of which doctrinal approach courts have used, judges have generally taken care to ensure that copyright law should not be interpreted to grant patent-like protection to program innovations. If no single approach to judging software copyright claims has

440. *Id*.

441. *Id*. at 1279. Venture-backed startups were more likely than other software firms to own or seek patents. *Id*.

442. *Id*. at 1292, 1303.

443. *See id*. at 1308–09.

444. *See* Pamela Samuelson, *The Uneasy Case for Software Copyrights Revisited*, 79 GEO. WASH. U. L. REV. 1746, 1776–81 (2011) (giving examples of developments in the software industry, such as the provision of software as a service instead of a product, that lessen the need for copyright protections to attain competitive advantage).

445. Google does not charge money for the installation of Android on mobile devices. *Oracle*, 750 F.3d at 1351.

446. *See, e.g.*, BAND & KATOH, INTERFACES ON TRIAL 2.0, *supra* note 10, at 45–46.

emerged, this is perhaps unsurprising given that copyright doctrines largely evolved to address issues posed by cases involving expressive works of art and literature, not functional processes such as programs. As *Altai* noted, programs are square pegs that courts must try to fit in the round holes of copyright.[447] But for the Supreme Court's *Baker v. Selden* decision, which established that copyright protects authorial expression, not functionality, courts would be floundering much more than they have.

What we can say with considerable confidence is that *Altai* and its AFC test for software copyright infringement were vast improvements over the *Whelan* framework and test for infringement under which every design decision a programmer made was protectable expression except for the rare elements as to which no alternative choices existed. *Altai* recognized that the utilitarian nature of programs meant that the scope of software copyright protection must necessarily be thin. *Altai* identified several categories of unprotectable elements of programs—efficient designs, externally constrained designs, elements common to works of that kind (i.e., scenes a faire elements) and public domain components—and directed that those elements be filtered out before assessing whether the defendant had infringed a software copyright. The main criticism this Article has levied against *Altai* concerned its failure to mandate, in addition, filtration of § 102(b) methods and processes.

Courts have sometimes applied the § 102(b) exclusions, with or without reference to the AFC test, when plaintiffs have alleged infringement based on copying of unprotectable methods or procedures. Among the elements of programs that have been adjudged unprotectable under § 102(b) are algorithms, program behavior, and command structures needed for achieving interoperability. The merger doctrine has complemented § 102(b) exclusions in numerous cases in which defendants used the necessary incidents to reimplement the same functionality as an existing program. This Article has called for explicit recognition of a merger of function and expression doctrine to supplement the merger of idea and expression and of fact and expression doctrines established in the case law. It has also recognized that copyright and patent law should play different roles in the legal protection of computer programs, even though the boundary lines between these laws, as applied to programs, has proven more elusive to articulate than in respect of other copyright subject matters. The Article concurs in an approach that recognizes that the patentability of some program innovations, as well as the unpatentability of abstract program

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

designs, should be taken into account in assessing the proper scope of copyright protection in software.

There is, of course, a simpler way to address software copyright infringement claims. It would begin by recognizing that copyright law can and does protect software developers against market-destructive appropriations of their products through the well-established rule that protects source and object code from illicit copying.[448] Similarly established is the protectability of audiovisual and other conventionally expressive content that may be displayed when computers are executing program instructions.[449] Making minor changes to program texts, such as changing variable names, rearranging instructions, or recompiling the code to disguise infringement should also not be tolerated. Translation of programs from one programming language to another or other forms of slavish copying may also qualify as infringement.[450] The main value of copyright protection for software developers lies in protecting these aspects of programs. If copyright law did nothing more than safeguard these literal and nonliteral elements of programs, the software industry would very likely still thrive. Limiting the scope of software copyrights to these protectable elements would likely make a special test for infringement of these works unnecessary.

Yet, courts have invested so much in articulating and applying tests for software copyright infringement that they may be reluctant to abandon these tests. Although the *Altai* AFC test has thus far been the most stable and widely accepted approach to judging software copyright infringement, several courts have adapted it. Some courts, for instance, have decided that the first step's construction of an abstractions hierarchy for the plaintiff's program is unnecessary when the claim of infringement is based on certain specific elements of the defendant's program.[451] Courts then proceed to consider whether various limiting doctrines of copyright law apply to those elements, in keeping with the *Altai* filtration step. In some cases, application

448. *See supra* text accompanying note 130; *see also* Weinreb, *supra* note 10, at 1250.

449. *See, e.g.*, Williams Electronics, Inc. v. Artic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982) (videogame graphics protectable).

450. The focus in most software infringement cases has been on the reproduction right, but modifying another firm's software and then selling the modified version to the public would also likely infringe the derivative work right. *See, e.g.*, Allen-Myland, Inc. v. IBM Corp., 746 F. Supp. 520 (E.D. Pa. 1990).

451. *See, e.g.*, Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1373 (10th Cir. 1997) ("Where, as here, the alleged infringement constitutes the admitted literal copying of a discrete, easily-conceptualized portion of a work, we need not perform complete abstraction-filtration-comparison analysis.").

of the limiting doctrines have made the third step unnecessary, for the plaintiff's claims failed because the doctrines precluded protection for the elements which the plaintiff claimed were her expressions.[452] Some courts have adapted the *Altai* second step to filter out procedures, processes, methods of operation, and systems.[453] Only rarely do courts specify which "golden nuggets" of expression remain after the filtration step, even though *Altai*'s third step had directed these elements of programs to be the starting point of the final step of assessing infringement.

I propose the following refinement of the *Altai* test for software copyright infringement. The first step would require the plaintiff to specify exactly which elements of her program that she alleges as the basis of the infringement claim. In most cases, it will be unnecessary to construct a hierarchy of abstractions for the program as a whole because only certain elements are alleged to infringe. A second step should inquire which, if any, allegedly infringing elements lie outside the scope of copyright protection under various limiting doctrines, such as the exclusion of (a) unoriginal elements; (b) abstract ideas, concepts and principles under § 102(b); (c) facts, know-how, and other public domain elements; (d) common elements for works of that kind, standard programming techniques, and constraints based on market demands under the scenes a faire doctrine; (e) efficient design elements; (f) command structures and APIs necessary for achieving interoperability with other programs or hardware; (g) other instances of merger of function and expression; and (h) procedures, processes, systems, and methods of operation under § 102(b). This would adapt the *Altai* AFC test to make its filtration step more rigorous, excluding a wider range of functional design elements in programs.[454]

---

452. *Id*. at 1376 ("In sum, although Mitel's values constitute non-arbitrary original expression, they are unprotectable as scenes a faire because they were dictated by external functionality and compatibility requirements of the computer and telecommunications industries.").

453. *See, e.g*., Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–43 (10th Cir. 1993).

454. Under this broader filtration analysis, Rand Jaslow might not have been held an infringer as to the dental lab program he developed to compete with Whelan. Jaslow certainly infringed Whelan's copyright when selling her program to third parties, but it is less clear, given how the law has evolved, that his competing program infringed. The courts in *Whelan* indicated that Jaslow did not directly translate Whelan's program from one language or another (which might justify the finding of infringement). *Whelan*, 797 F.2d at 1228 ("Dr. Moore testified that although the Dentcom program was not a translation of the Dentalab system, the programs were similar in three significant respects."). Had the AFC test been applied in *Whelan*, the file structures Jaslow used, for instance, may have been efficient for the tasks at hand. The court in *Altai* rejected an overall structural similarity claim on scenes a faire grounds, which might have been true in *Whelan* also. The

In some cases, as in *Lexmark* and *Borland*, there will be no need for a third step because the filtration step results in a judicial conclusion that even if the defendant copied something from the plaintiff's work, the expressive elements of the plaintiff's program were not copied. Yet in other cases, courts should proceed to consider whether the defendant copied a sufficient quantum of expression from the plaintiff's work to be held as an infringer. Adoption of this refined *Altai* test would be consistent with the overwhelming majority of software copyright cases and with longstanding principles of U.S. copyright law. It would also promote competition and ongoing innovation in the software industry, in keeping with the constitutional goal of copyright of promoting progress in science and useful arts.

The CAFC's *Oracle* decision obviously took a very different approach. This Article has shown that the CAFC's *Oracle* misinterpreted *Altai*, misunderstood § 102(b), and misapplied the merger doctrine, as well as major court rulings that have applied these rules. The *Oracle* decision is contrary to Supreme Court rulings and to the CAFC's own precedents, especially in treating utility patent and copyright laws as providing overlapping protections for computer program innovations.[455] The CAFC also failed to grasp that an API that specifies functions that a program is designed to carry out is fundamentally different from the copyright-protectable program that implements that API in independently written code.

Some may think that the *Oracle* decision, erroneous as it is, will have little impact on subsequent cases. The decision is, after all, an outlier in the case law and involves complicated facts. Software copyright cases will, moreover, generally go to the regional circuits, not to the CAFC, which has no jurisdiction in copyright cases except when there is a patent claim in the case. But *Oracle* has introduced new uncertainties in the law of software

---

similarities in the operation of certain modules—order entry, invoicing, accounts receivable, end of day procedure, and end of month procedure—appear to be automations of Jaslow's business processes that should have been filtered out under § 102(b). *See id.*

455. The CAFC remanded *Oracle* for retrial of Google's fair use defense. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1372–77(Fed. Cir. 2014). It concluded that "this is not a case in which the record contains sufficient factual findings on which we could base a de novo assessment of Google's affirmative defense of fair use." *Id.* at 1377. It also indicated that compatibility considerations could be considered in the fair use context. *Id.* at 1376–77. The CAFC characterized Oracle's argument for granting summary judgment on fair use as "not without force," but ultimately concluded that a new trial was needed. *Id.* On remand, a jury found Google's reimplementation of the Java APIs constituted fair use. Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA, 2016 WL 3181206 (N.D. Cal. June 8, 2016), *appeal docketed*, Nos. 17-1118, -1202 (Fed. Cir. Nov. 14, 2016).

copyrights, and software companies now have reasons to litigate to test the CAFC's resurrection of the *Whelan* approach to assessing infringement. Given how many software patents are out there, it may be easy for a plaintiff's lawyer to find one in a client's portfolio or for the client to buy one so that the complaint may allege a defendant infringed patents as well as copyrights.[456] In such cases, the appeals would go to the CAFC instead of to a regional circuit.

In the aftermath of *Alice* and court decisions striking down software patents, it may be tempting for courts to interpret the scope of copyright protection expansively, as the CAFC did in *Oracle*, because the role of patents in protecting program innovations is receding. Under a kind of conservation of IP incentives theory, copyrights might seem to need to be broader to make up for the fact that patents are providing less protection for program innovations. Copyright does important work in protecting programs, but as the Second Circuit recognized in *Altai*, "fundamental tenets" of this law should not be distorted to fill a perceived gap in legal protection for programs.[457]

If software developers need some additional legal protection for industrial design elements of programs that neither copyright nor patent law can provide, they should take their case to fill that gap to Congress, not use the courts to fill the gap through expansive interpretations of copyright protections.[458] Sui generis forms of protection for software have been proposed in the past, and perhaps this approach should be reconsidered.[459] Industrial design laws typically provide a relatively short term of protection against market-destructive appropriations.[460] At present, however, competition and innovation seem to be thriving in the software industry without additional legal protections. Without substantial evidence to

456. *See supra* note 30 (discussing cases subsequent to *Oracle* in which plaintiff's lawyers have learned this lesson).

457. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992) ("While incentive based arguments in favor of broad copyright protection are perhaps attractive from a pure policy perspective, ultimately, they have a corrosive effect on certain fundamental tenets of copyright doctrine." (citation omitted)).

458. *See id.* ("[T]he resolution of this specific issue could benefit from further legislative investigation . . . .").

459. *See, e.g.*, Samuelson et al., *Manifesto*, *supra* note 10, 2405–20 (proposing a short term of sui generis protection for the industrial compilations of applied know-how embodied in computer programs).

460. *See, e.g.*, Reichman, *supra* note 1, at 2459–65 (discussing European industrial design laws).

support a change in the law,[461] the courts should continue to apply copyright protections for software in keeping with *Altai*, its progeny, and traditional principles of copyright law.

461. *See, e.g.*, Robert Kastenmeier & Michael Remington, *The Semiconductor Chip Protection Act: A Swamp or Firm Ground?*, 70 MINN. L. REV. 417, 439–42 (1985) (evidence of industry needs, among other things, should inform expansions of IP rules).