Journal of Intellectual Property Law

Volume 24 | Issue 1 Article 3

October 2016

Oracle v. Google and the Scope of a Computer Program Copyright

Dennis S. Karjala
Arizona State University School of Law

JOURNAL OF INTELLECTUAL PROPERTY LAW

VOLUME 24 FALL 2016 NUMBER 1

ARTICLES

ORACLE V. GOOGLE AND THE SCOPE OF A COMPUTER PROGRAM COPYRIGHT

Dennis S. Karjala[*]

TABLE OF CONTENTS

[*] Jack E. Brown Professor of Law, Arizona State University.

I. INTRODUCTION

Copyright law protects the expression of an idea but not the idea expressed. Everybody knows that. The problem has always been that nobody knows or can articulate what parts of a work constitute "expression" and what parts constitute "idea." For traditional copyright subject matter like books, art, and music, courts and scholars typically look to the "creative" aspects of the work and the availability of a variety of ways of expressing the same idea.[1] Decisions are largely ad hoc[2] and nuances have developed depending on the type of work at issue.[3]

Many courts have addressed the problem of determining the scope of copyright protection in a computer program in the same ways they have long used for traditional copyright subject matter.[4] In fact, however, computer programs offer at least one example in which copyright doctrine, statutory language, and fundamental notions of intellectual property policy all lead to a clear and consistent result that gives less protection to program structure than, say, copyright in a fictional story: Copyright in a computer program should be limited to literal program *code* and close paraphrases of program code. The reason is that computer programs are technological subject matter masquerading as copyright-protected literary works.[5] Intellectual property law has always treated art and technology differently, and maintaining patent's general role in protecting technology while accomplishing the anti-misappropriation congressional purpose of placing program code under copyright can only be achieved by appropriate limitation of the scope of the program copyright.[6]

[1] Montgomery County Ass'n of Realtors, Inc. v. Realty Photo Master Corp., 878 F. Supp. 804, 810 (D. Md. 1995), *aff'd per curiam*, 91 F.3d 132 (4th Cir. 1996) (marketing "puffery" added to multiple real estate listings is not factual and supplies the necessary creativity for copyrightability); Stern v. Does, 978 F. Supp. 2d 1031, 1041 (C.D. Cal. 2011) (copyright protectability of a very short textual work depends on the presence of creativity).

[2] *E.g.*, Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) ("The test for infringement of a copyright is of necessity vague [and] . . . [d]ecisions must therefore inevitably be ad hoc.").

[3] *See infra* notes 39–40 and accompanying text.

[4] Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir. 1986) might be thought of as the beginning of this approach. The district court decision in Lotus Development Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990) is another. *Lotus v. Paperback* is critically analyzed in notes 42–44 and accompanying text *infra.*

[5] David G. Luettgen, *Functional Usefulness vs. Communicative Usefulness: Thin Copyright Protection for the Nonliteral Elements of Computer Programs*, 4 TEX. INTELL. PROP. L.J. 233, 260 (1996).

[6] Dennis S. Karjala, *Copyright Protection of Computer Software, Reverse Engineering, and Professor Miller*, 19 U. DAYTON L. REV. 975 (1994) (explaining anti-misappropriation is the sole justification

II. COMPUTER PROGRAMS ARE TECHNOLOGY

Computer programs are the *technology* that causes computers to perform their functions.[7] The word "technology" is important, because the now-centuries-long dogma of copyright law developed for the non-technological subject matter of art, literature, and music.[8] With its low threshold for and very long period of protection, copyright is relatively generous with the exclusive rights it affords to copyright owners. Patent law, on the other hand, has always provided for the possibility of exclusive rights in technology but is stingier to technology developers. Patents require formal applications and examination, precisely worded claims, and, even if issued, expire only twenty years after filing.[9] Congress's decision to break from tradition in the 1976 Copyright Act by bringing computer programs under copyright protection—without explicit recognition that such a break from tradition was being made—can only be sensibly interpreted in the light of the long history of Intellectual Property Law in which copyright protected nonfunctional works of information and patent protected functional works of technology.[10] Copyright doctrine (idea/expression, substantial similarity, fair use) is notoriously vague, and valid doctrinal arguments can be mustered for either a broad or a narrow scope of copyright protection in computer programs. Without reference to policy, therefore, especially the policy underlying the patent and copyright subject matter distinction, no argument is determinative. Recognition that we have *two* statutes for protecting the fruits of intellectual creativity that operate in now overlapping but fundamentally different spheres, however, leads naturally and almost inexorably to a clear resolution: Copyright for computer programs protects the literal program code; beyond program code, the program copyright protects, at most, close paraphrases of literal code. The program copyright does not protect interfaces generated by the program code, nor should it protect the variety of technological means by which the program code is structurally organized.

for bringing computer programs—functional works of technology-under copyright). *See also* Note, *Copyright Protection of Computer Program Object Code*, 96 HARV. L. REV. 1723, 1741 (1983) (stating "The structure of the 1976 Act itself indicates that the Act can be understood to function primarily as an anti-misappropriation statute").

[7] The statutory definition of "computer program" is a set of statements or instructions to be used in a computer to achieve a certain result. Bringing about results in the real world is essentially the definition of technology.

[8] *See infra* note 12.

[9] 35 U.S.C. 154(a)(2).

[10] *See* Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 CONN. L. REV. 439, 448–58 (2003).

Probably the most important notion to keep in mind in determining the scope of computer program copyrights is that we have *two* major intellectual property statutes—patent and copyright.[11] Traditional copyright law applied to nonfunctional art, music, and literature, while traditional patent law applied to functional works of technology.[12] That to some extent we now protect technology in the form of computer programs under copyright law does not and cannot mean, until Congress instructs otherwise, that all of the protection we have recognized for nontechnological novels and plays must carry over in full force to computer program technology.

A second point, grounded in both policy and doctrine, is that "creativity" alone does not equate to copyright-protected "expression."[13] A third and related point is that the availability of a wide range of choices—"absence of merger"—does not equate to copyright protection for any particular choice.[14]

A fourth point is that computer programs, by their nature, are methods of operation[15] (they are quite literally the technology for operating computers to bring about specific results). Any particular program is a made up of series of steps that achieve the result effected by that program. It necessarily follows that, given Congress's near verbatim adoption of the CONTU[16]

---

[11] Copyright Act of 1976, 17 U.S.C.A. (West 2012); Patent Act of 1952, 35 U.S.C.A. (West 2012).

[12] The word "functional" is used in many ways in the context of copyright law, so it is important to be clear: "Functional" in the sense that distinguishes patent and copyright subject matter does *not* mean merely "useful," any more than the definition of a "useful article" in the Copyright Act refers to everything that is useful. A "useful article" under copyright is something that has "an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information." 17 U.S.C.A. § 101 (definition of "useful article"). Maps are useful, but they are not useful articles, because they do no more than convey information. *See* Karjala, *supra* note 10, at 448–58. While the copyright definition of "useful article" is explicitly applied by the statute only to "pictorial, graphic, and sculptural" works, the underlying distinction between works having an intrinsic utilitarian function *other than* to supply information to human beings and works that do not captures most of the historical distinction between patent and copyright subject matter. *See* Luettgen, *supra* note 5, at 252–55 (discussing the role of patent law in protecting function and the limited role of copyright).

While not articulated in identical reasoning, David G. Luettgen has come to a similar conclusion of the role of patent law and protecting function and the limited role of copyright. Luettgen, *supra* note 5, at 256.

[13] *Feist* says that protection requires a modicum of creativity.

[14] Dennis S. Karjala, *Copyright Protection of Computer Program Structure*, 64 BROOK. L. REV. 519, 521–22 (1998); Dennis S. Karjala, *A Coherent Theory for the Copyright Protection of Computer Software and Recent Judicial Interpretations*, 66 U. CIN. L. REV. 53, 91 (1997) [hereinafter Karjala, *A Coherent Theory*].

[15] We could add that they are "systems," "procedures," and "processes" as well, but the analysis does not depend on which term we use for classification.

[16] Congress established a commission, CONTU, to make recommendations on software protection and other matters in 1974. Final Report of the National Commission on New Technological Uses of Copyrighted Works 4 (1978) [hereinafter CONTU Report]. *Id.* at 15

recommendations, § 102(b) has been limited in its application to computer programs. The question is, how far does this limitation go—does § 102(b) not apply to computer programs at all, or is the limitation itself more limited in applying mainly, if not solely, to program code?

A fifth point arises directly from the definition of a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[17] This definition expressly distinguishes between the code (the set of "statements or instructions") and the "certain result"[18] brought about by that set of instructions. It is the set of statements or instructions that constitutes the computer program. The "certain result" brought about by execution of those instructions in a computer is not a part of the computer program and, logically, is not protected by the copyright in the computer program. Some such "certain results," of course, will be independently copyright protectable as, for example, pictorial, graphic, or literary works,[19] but such copyrights exist independent of the copyright in the computer program itself.

Sixth, and finally, even if we were to apply traditional copyright doctrine for literary works to computer programs without regard to their sui generis nature as works of technology, we should *still* conclude that the program copyright covers primarily, if not solely, the program code and certainly not program structure, sequence, and organization (SSO), let alone the functional interfaces generated by the program code. Some traditional literary works have a broad scope of protection (novels and plays), but others have a very narrow scope of protection—protection that is limited virtually to their verbatim language (insurance contracts).[20] Anyone demanding computer programs be given the broad scope of protection afforded to novels and plays should supply a policy-based reason for analogizing computer programs to such works, as opposed to

(concluded that patent and trade secret law were inadequate to protect computer programs because notwithstanding program vulnerability to quick and exact copying, most programs lacked the "nonobvious" technological advance required by patent. Further, many were publicly distributed, removing them from trade secret status). *Id.* at 17 (concluding also that computer programs were already protected under copyright under the broad definition of original works of authorship adopted by the 1976 Act). Consequently, the Commission recommended adoption of a specific definition of a "computer program" and a new § 117 to provide for copying of programs as a step in their use or for archival purposes. *Id.* at 12. *See also* Luettgen, *supra* note 5, at 256–57.

[17] 17 U.S.C.A. § 101 (West 2012) (definition of "computer program").

[18] *Id.*

[19] Computer Assocs. Int'l Inc. v. Altai Inc., 982 F.2d 693, 703 (2d Cir. 1992) (noting that things like screen displays "represent products of computer programs, rather than the programs themselves").

[20] Continental Casualty Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958).

more technical works like dictionaries, legal forms, and scientific works in which the scope of protection is much narrower.[21]

The ineluctable conclusion that must be drawn from these basic points is that computer programs are a sui generis type of copyright subject matter (even though falling within the classification of "literary works"). The program copyright covers the program code and close paraphrases of such code, for the purpose of preventing cheap, rapid and exact copying. However, the copyright does not extend to functional aspects of program structure; it does not extend to *any* aspect of program structure, because all such structure is intended to achieve a functional result in an efficient way.[22] A fortiori, it does not extend to any aspects of the program interfaces, which exist at an even higher level of abstraction than the program SSO.[23]

This interpretation gives full copyright protection to that aspect of program technology that is most in need of protection from verbatim copying, namely, electronic representations of program code.[24] It leaves other aspects of program technology to develop under the rules of patent and trade secret, which are designed for the protection of technology. No one has ever offered a compelling policy-based reason for treating noncode program technologies like SSO differently from any other technology, for which patent and trade secret have been the sole choice for centuries, and Congress has given no hint, let alone explicit direction, that copyright protection for technology should extend so far beyond computer program code.

III. *ORACLE V. GOOGLE*

The need to consider the sui generis nature of computer programs as copyright subject matter has been brought into stark relief by the conflicting district court and Federal Circuit decisions in *Oracle America, Inc. v. Google, Inc.*[25] At issue were the names or headings of some thirty-seven programs constituting part of the Java programming system in which Oracle was the

[21] Karjala, *A Coherent Theory*, *supra* note 13, at 81.

[22] Because no one but an expert can even find, let alone use, electronic form of code, there is no need to put anything other than functional code into the program. No sane programmer would put esthetically pleasing but nonfunctional code (assuming that such a thing exists) into the program.

[23] As previously mentioned, interface elements may be independently protected as works of another type, such as pictorial or graphic works, but that has nothing to do with the program copyright.

[24] Luettgen, *supra* note 5, at 258–60.

[25] 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014). The action as originally filed included patent claims, so even though no patent questions were at issue on appeal, jurisdiction was in the Federal Circuit; *id.* at 1353.

copyright owner.[26] Google believed that the functionality of these thirty-seven programs would be useful to programmers of applications for its mobile telephone Android operating system and wrote independent programming code to accomplish these functions. However, the headings (or "declarations") used by programmers to call the various functions into operation were identical in the Google system to the headings used in Java corresponding to these same functions.[27] Moreover, the individual programs (called "methods") were organized into similar or identical "classes" in Android as in Java.[28] The district court applied copyright doctrine narrowly with an express eye on the underlying policy that functional invention is protectable only under patent law.[29] The Federal Circuit applied copyright doctrine broadly on the assumption that Congress's decision to place functional computer programs under copyright as a "literary work" meant, essentially, that technological functionality was not a limitation on the scope of a program copyright.[30]

The reason for the dramatically contrasting approaches taken by the district and appellate courts in the *Oracle* case likely inheres in the unnecessarily confusing approaches taken by the two cases generally regarded as the leading authorities for the scope of a program copyright and its user interfaces. Those cases reached results that comport with the functional nature of computer programs, but they were not reasoned in such terms.[31] The results in each case were that computer program *code* is protected by the program copyright but that structural elements of programs—by their nature designed for efficient operation—are relegated to their fate under patent (and trade secret) law.[32] Moreover, functional interface elements are also not protected by the program copyright and must find copyright protection, if at all, as one of the more traditional classes of copyright subject matter, such as pictorial or graphic works.[33] This Article seeks to explain why these results are correct and show both where the Federal Circuit went wrong and to illuminate a policy-based

[26] *Oracle Am., Inc.,* 872 F. Supp. 2d at 975.

[27] *Id.* at 998.

[28] *Id.* at 979.

[29] *Id.* at 977.

[30] 750 F.3d at 1367 (construing 17 U.S.C. § 101 (2010)).

[31] Computer Assocs. Int'l, Inc. v. Altai, 982 F.2d 693 (2d Cir. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

[32] *Computer Assocs. Int'l, Inc.*, 982 F.2d at 693; *Lotus*, 49 F.3d at 807, *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

[33] The court in *Borland* found the menu command hierarchy to be an unprotected "method of operation" under section 102(b). If the program copyright does not cover interfaces that are methods of operation, the only potential source of copyright protection would be traditional copyright subject matter under section 102(a).

path for the future protection of computer programs under copyright that is perfectly compatible with standard copyright doctrine, the statutory language, and fundamental intellectual property policy.

IV. Patent and Copyright

As discussed briefly above, while both patent and copyright give exclusive rights in the fruits of intellectual creativity, they go about their business in very different ways. Patent is stricter than copyright, requiring a formal application that is reviewed by a skilled examiner, a precise delineation in the claims of what is to be protected by the patent, and a nonobvious advance in the technological art. The exclusive rights of even the successful patent applicant, moreover, endure only from issuance until twenty years from the date the patent application was filed. Copyright in a work of authorship, on the other hand, arises automatically upon fixation, and the copyright owner need not specify the scope of protection in the work—what parts constitute protected "expression" and what parts are unprotected "idea." Moreover, copyright endures for a very long time—life of the author plus seventy years.[34]

The only way to account for these very different ways of achieving a similar result is the nature of the subject matter with which these two statutes are intended to deal.[35] Patent has always been aimed at the protection of functional technology,[36] which typically advances in an incremental fashion. Incremental "improvement" of a copyright-protected work necessarily infringes as substantially similar, meaning that the copyright owner has the sole right to make incremental improvements for the very long period of copyright protection. For nonfunctional works like novels and plays, this perhaps does

[34] Copyright Act of 1976 § 302(a); 17 U.S.C.A. § 302(a) (West 2012).

[35] Karjala, *A Coherent Theory*, *supra* note 14, at 56–66.

[36] It is crucial to understand the word "functionality," as the primary distinction between patent and copyright subject matter, with precision. It does *not* mean something that is merely "useful." Much of the distinction between the two types of subject matter is contained in the Copyright Act's definition of a "useful article": something that has "an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information." 17 U.S.C.A. § 101 (West 2010) (defining "useful article"). Thus, maps and dictionaries, while often quite "useful," are not useful articles. Nor are they functional in the sense used to distinguish patent and copyright subject matter, and of course they are copyright and not patent subject matter. *See* Karjala, *A Coherent Theory*, *supra* note 14, at 57–58. A more general approach to the distinction between patent and copyright subject matter starts from this same point but looks to the notion of "incremental improvability" to capture aspects of patent subject matter, such as processes, that the useful article definition does not quite reach. *See* Karjala, *supra* note 10, at 448–68. This more general definition of "functionality" is unnecessary for resolution of the issues in the *Oracle* case.

little harm, at least to technological development. For functional technology, however, it would almost surely have a net inhibiting effect on technological progress. It is difficult even to imagine how the auto industry would have developed had cars been objects of copyright protection. Hence, the relative stinginess of patent law.

V. APPLICATION TO PROGRAMS

Computer programs are technology, yet Congress has directed their protection under copyright law. Any reading of the CONTU Report makes it clear that the Commission was concerned with the vulnerability of computer program *code* to easy and rapid misappropriation.[37] Program code is essentially the only aspect of program technology that the CONTU Report explicitly references.[38] Program code is technological subject matter, so, to that extent, we must assume that by adopting the CONTU recommendation, Congress intended to place technological subject matter—traditionally the subject of patent law—under copyright protection. CONTU, however, said nothing about extending the scope of copyright protection in program code to so-called "non-literal elements" like "structure, sequence, and organization" (SSO) or program interfaces, let alone programming languages. SSO, interfaces, and programming languages are all technological in nature. They are designed, created, and used to make better computer programs, which themselves are the technology for using computers. Because CONTU, and therefore Congress, did not consider these non-literal elements of computer programs, and because no one has ever shown that any of them are any more subject to piracy than other types of patent subject matter, courts should hesitate in casually allowing copyright to trench so deeply into the realm of patent. If patent law does not protect some of these technological advances, that is a matter of *patent* policy. There is no need or basis for courts to conclude, absent congressional authorization, that the very long term and broad protective scope of copyright law should be applied to these technological advances just because they are ineligible for patent protection (if that should be the case). As discussed below, neither; classification as a "literary work," the existence of "creativity" in program design, nor the availability of a wide range of design choices can lead to copyright protection for any given choice.

[37] *See* CONTU Report, *supra* note 16, at ch. 3; *see also*, *e.g., id.* saying "A computer program may be misappropriated in a variety of ways. In the first and most straightforward instance, the program listing or the programmer's original coding sheets . . . ."

[38] *Id.*

Courts should not be misled by the formal classification of computer programs as "literary works." While the scope of protection for fictional works like novels and plays is relatively broad,[39] other literary works, like insurance contracts, have a very narrow scope of protection—essentially limited to verbatim language.[40] For computer programs, Congress placed technological subject matter under copyright law without specifying the intended scope of protection. An interpretation that gives meaningful application to the congressional directive while minimally distorting the traditional balance between patent and copyright subject matter is to limit the program copyright to program code and close paraphrases of code. To go beyond such an interpretation by protecting SSO and other noncode functionality risks giving the long term and vague scope of copyright protection to technology without congressional directive to do so.

"Creativity" in program design is not itself a justification for copyright protection. A fundamental error made by the district court in *Lotus v. Paperback Software International*[41] was adopting the notion that the creative aspects of a computer program are the aspects Congress intended to protect *under copyright law*.[42] While creativity is a *necessary* condition for copyright intended to protect

[39] *E.g.*, Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49 (2d Cir. 1936) (finding infringement of the copyright in a play by the overall plot of a movie, notwithstanding different names, places, and times).

[40] *E.g.*, Cont'l Cas. Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958).

[41] 740 F. Supp. 37 (D. Mass. 1990).

[42] *E.g.*,

> [T]he bulk of the creative work is in the conceptualization of a computer program and its user interface, rather than in its encoding, and . . . creating a suitable user interface is a more difficult intellectual task, requiring greater creativity, originality, and insight, than converting the user interface design into instructions to the machine. . . . Defendants' contentions would attribute to the statute a purpose to protect only a narrowly defined segment of the creative development of computer programs, and to preclude from protection even more significant creative elements of the process. Such a result is fundamentally inconsistent with the statutory mandates.

Id. at 56. Later, the same court said,

> [T]he more innovative the expression of an idea is, the more important is copyright protection for that expression. By arguing that 1-2-3 was so innovative that it occupied the field and set a de facto industry standard, and that, therefore, defendants were free to copy plaintiff's expression, defendants have flipped copyright on its head. Copyright protection would be perverse if it only protected mundane increments while leaving unprotected as part of the public domain those advancements that are more strikingly innovative.

Id. at 79.

under copyright law.[43] However, creativity cannot be a *sufficient* condition for copyright protection. Everyone agrees that a creative idea, like the theory of relativity, is not protected by the copyright in the scientific paper in which the theory is announced. It may be that the creative idea is the most important part of the paper—the part in which the creative genius of the author most brilliantly shines through. Selden's methods of bookkeeping were likely the most creative aspect of his new system, but that did not and cannot imply that Selden's creative choices give him exclusive rights in the accounting system he explained in his copyright-protected book.[44] Creativity has never been a basis for copyright protection of an idea, and that remains the case whether or not a patent is available for the idea.[45] If a patent is unavailable, that is a matter of *patent* policy and certainly, even if patent policy is in error, not something that should "corrected" by longer-term and easier-to-obtain exclusive rights under copyright. Thus, if an aspect of a computer program code or interface is an idea, system, or method of operation, no amount of creativity in the design of that aspect leads to protection by the program copyright.

Nor can the availability of a wide range of choices—"absence of merger"—equate to copyright protection for any particular choice. There is a wide range of choice for methods of bookkeeping, but that did not and cannot imply that Selden's particular choice gives him exclusive rights in the creative system he explained in his copyright-protected book.[46] Until Congress has spoken, as it has for computer program code, if something like an accounting system is excluded from copyright protection under § 102(b), it remains unprotected no matter how many alternatives are available, theoretically or otherwise. This makes perfect sense once the distinction between patent and copyright subject matter, functional technology versus nonfunctional art, is recognized. Technology tends to move, via incremental improvement, toward an optimal solution to a given problem.[47] Indeed, Selden's was only the most recent step in the gradual development of accounting over centuries.[48] If a previous copyright owner of an accounting system has the exclusive right to make improvements, there would be little competition among would-be improvers,

[43] Feist Pubs., Inc. v. Rural Tel. Serv. Co., 499 U.S. 340 (1991). *But* see Dennis S. Karjala, *Copyright and Creativity*, 15 UCLA ENT. L. REV. 169, 174–79 (2008) (critiquing Feist Pubs. Inc. v. Rural Tel. Serv. Co.).

[44] Baker v. Selden, 101 U.S. 99, 107 (1880).

[45] Karjala, *supra* note 43, at 173.

[46] *Baker*, 101 U.S. at 107.

[47] Karjala, *supra* note 43, at 180.

[48] *See* Luca Paciola, *Summa de Arithmetica, Geometria, Proportioni et Proportionalita* (1494), *in* T. BUDD & E. WRIGHT, THE INTERPRETATION OF ACCOUNTS 7 (1930).

not just for the twenty years afforded by patent law, but for the much longer term of copyright.

The problem for courts is that copyright is supposed to protect "expression," and no one has ever been able to come up with a definition that operationally distinguishes unprotected ideas from protected expression.[49] For traditional copyright subject matter, the ad hoc nature of judicial approaches might be disconcerting to specific parties, but they do not result in affording technological subject matter the long term of copyright protection.[50] Using some inchoate judicial notion of "creativity," therefore, may be the best that courts can do for works like novels, plays, and movies. Technological creativity, however, requires more care, lest courts find themselves protecting technological aspects of computer programs under copyright that would not even be eligible for the twenty years of protection afforded by patent law.

For computer program code, copyright protection is mandated by Congress. Courts are quite correct in protecting verbatim and near-verbatim code under copyright, notwithstanding code's functional nature and notwithstanding its (correct) characterization as a "process" or "method of operation" under section 102(b).[51] No one, however, has offered a justification for taking *noncode* aspects of computer programs out of their technological home in patent law. The broad scope of protection for traditional literary works, like novels and plays, does not suffice.

As discussed above, outside of the CONTU Report itself, there is no legislative history concerning the 1980 amendments adopting the CONTU recommendations. CONTU itself said very little about its intended scope of protection. It is clear that CONTU was thinking primarily about program code. The Report does not mention the protection of interfaces, and the closest it gets to what is now termed "SSO" is the flow chart. CONTU refused to draw a line between the flow chart and the source code in favor of a more general focus on the distinction between idea and expression, which it defined to be the distinction between the writing and the process described by the writing.[52] The

[49] *E.g.*, Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Nobody has ever been able to fix that boundary, and nobody ever can.").

[50] *E.g.*, Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) (stating "The test for infringement of a copyright is of necessity vague. . . . Decisions must therefore inevitably be ad hoc.").

[51] 17 U.S.C. § 102(b) (2012).

[52] CONTU Report, *supra* note 16, at 25. Even at the time the CONTU Report was written, there was no serious question about the copyright protectability of program flow charts. Technical drawings like circuit diagrams have long been considered copyright subject matter. The *scope* of protection in, say, a circuit diagram was of course very narrow, and did not extend to the functionality generated by combining physical electronic devices according to the diagram. Only

question, therefore, remains whether Congress, in taking computer programs outside the limitations of § 102(b), intended this exception to be limited to program *code* (the only program element that is subject to quick and exact copying) or to other program aspects such as SSO or user interfaces.

Any complex computer program can be written in an extremely large number of ways, many of which are likely to be of roughly equal efficiency—whatever definition of "efficiency" we use for this purpose. Protection of each individual literal code version of accomplishing the result achieved by the program, therefore, does not substantially inhibit subsequent programmers seeking to achieve the same result, and it does protect against unscrupulous electronic copying by those seeking to gain a competitive advantage by avoiding the development costs incurred by the first programmer. It is not "absence of merger" as such, however, that justifies the copyright protection of code. That justification inheres in the vulnerability of code, especially in its electronic form, to quick and easy piracy. Neither SSO nor interfaces are similarly vulnerable, because they cannot be copied without undertaking the difficult task of reverse engineering the program and writing and testing independent code to achieve the desired function.

Doctrinal analysis, even under traditional copyright, law leads to the same conclusion. As examined above, the policy of protecting technology under patent law argues for limiting copyright protection to that functional computer program element that is vulnerable to piracy, namely, program code.[53] Traditional doctrine relating to the scope of protection of literary works would reach SSO only if computer programs are analogized to broadly protected literary works like novels and plays. If computer programs are analogized to technical works, scientific works, rule books, and legal forms, however, the scope of protection even under traditional copyright doctrine would be limited to near verbatim copying, that is, to literal program code.[54] No one has offered an argument, let alone a convincing argument, for treating a computer program more like a novel than a technical work.

> The statutory definition of a "computer program" gives powerful support to the doctrinal argument. The statute defines a computer program as "a set of statements or instructions to be

verbatim or near verbatim copying of the diagram, as a diagram, infringed. *E.g.*, Nat'l Med. Care, Inc. v. Espiritu, 284 F. Supp. 2d 424, 435–37 (S.D. W. Va. 2003) (holding that an "as-built structure," such as a cabinet, "cannot be an infringing copy of a technical drawing" and that technical drawings are thinly protected works that are infringed only by nearly exact copies).

[53] *See supra* note 4 and accompanying text.

[54] *E.g.*, Cont'l Cas. Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958).

used directly or indirectly in a computer in order to bring about a certain result."[55]

Few courts have incorporated this definition into their analyses, but it is crucial in analyzing the copyright protection of interfaces in particular and nonliteral elements in general. Interfaces are part of the "certain result" that the program code brings about: a program is written in such a way that it accepts certain inputs in defined formats and generates outputs in certain formats, often interactively with the user.[56] Those required input and output formats are the user interface. The headings or declarations that Google copied exactly for the Android system are repeatedly referred to by the Court of Appeals as "code," but in fact they become actual computer code only as part of an application program written in the Java (or Android) language, typically by third-party programmers.[57] The Java compiler (or Java "virtual machine") is programmed at a lower level of abstraction from actual source code written in Java.[58] This compiler is able to recognize the digital representations of instructions appearing in Java applications, just as Borland's spreadsheet program, operating in the Lotus compatibility mode, was able to recognize digital representations of the Lotus instructions and their command structure.[59] Google was not accused of copying code from the Java compiler, however, any more than Borland was accused of taking code from the Lotus program.[60] What was taken in both cases was not the *code* but the *products of code*—parts of that "certain result" brought about by execution of the instructions comprising the code. These were "taken" only in the sense that independent code was developed to achieve the same result.

The headings and declarations, therefore, are not protected by Oracle's copyright in the Java Virtual Machine software or related computer programs.

[55] 17 U.S.C. § 101 (2012) (defining "computer program").

[56] Karjala*, A Coherent Theory*, *supra* note 14, at 96–99; Elizabeth G. Lowry, Comment, *Copyright Protection for Computer Languages: Creative Incentive or Technological Threat?*, 39 EMORY L.J. 1293, 1303 (1990), argues that Lotus's claim to copyright the menu command hierarchy is an attempt to copyright a command language. She goes on to argue that programming languages are not copyright subject matter because they are systems of communication barred from copyright protection by section 102(b). *Id.* at 1335.

[57] *E.g.*, Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1352, 1356–57, n.4 (Fed. Cir. 2014).

[58] There may be various levels of programming within the compiler, but it is probably sufficient to think of the compiler as being programmed in what we used to call "machine language." *See also* Peter J. Denning & Robert L Brown, *Operating Systems*, 251 SCI. AM. 94 (1984) (detailing the many layers in which operating systems are designed).

[59] *Compare Google*, 780 F.3d at 1348, *with* Lotus Development Corp. v. Borland Int'l Inc., 49 F.3d 807, 810 (1st Cir. 1995).

[60] *Compare Google*, 780 F.3d at 1351, 1353, *with Lotus*, 49 F.3d at 810.

If protected by copyright at all, such headings and declarations need to qualify independently as a copyright protected work. Like the "menu command hierarchy" at issue in *Lotus v. Borland*, it is far from clear what kind of work that might be, even if we put functionality considerations to the side.[61] The set of headings and declarations is a set of functions that can be invoked in the Java language. The formats for each instruction are a system for invoking those available functions. There was almost surely a wide range of choice for the names of these functions and their formats, but that was true for the commands and the menu command hierarchy in *Lotus v. Borland* as well.[62] To say that Oracle has exclusive rights to the names of these functions and their formats would imply that the creator of any new programming language has exclusive rights in the names for the functions allowed by the language and the formats for invoking them. Thus, no one, without authorization, could take a subset of the allowed instructions in that programming language as the basis for a new programming language. An increasingly important branch of technology would then be protected for much longer than twenty years offered for other technological advances without a showing that the new set of available names and functions would have been eligible for a patent.

What is really at issue is thus the copyright protection of a programming language.[63] A programming language is a complex set of possible instructions and associated syntax that can be combined to cause a computer to achieve particular results.[64] A computer *program*, by statutory definition, is a particular set of the allowed instructions available in a given programming language that achieves a "certain result" upon execution by the computer.[65] A programming language, then, is not a computer program but rather a tool for writing computer programs. Such languages are functional, both in the technological sense that a particular language may lead to programs that are faster to code or compile, easier to debug or modify more economical in memory usage, or less prone to errors in coding in particular applications, and in the sense that their

[61] *See Lotus*, 49 F.3d at 811.

[62] *See, e.g.*, *id.*

[63] *See, e.g.*, Lowry, *supra* note 56, at 1293; Henry H. Perritt, Jr., *Unbundling Value in Electronic Information Products: Intellectual Property Protection for Machine Readable Interfaces*, 20 RUTGERS COMPUTER & TECH. L.J. 415, 448–49 (1994) (commands and syntax acceptable by a particular program constitute a programming language); Richard H. Stern, *Copyright in Computer Programming Languages*, 17 RUTGERS COMPUTER & TECH. L.J. 321, 327 (1991) (defining programming language as "a formal system of expression including. . . : (1) a set of vocabulary elements; (2) a set of syntax rules," and "the assignment of meaning to statements that properly combine vocabulary elements in accordance with syntax rules").

[64] Perritt, *supra* note 63, at 499.

[65] 17 U.S.C. § 101 (2012).

usefulness goes well beyond their "appearance" (or what they "say" to a human reader) to the accomplishment of work in the form of computer operation.[66] Different programmers may prefer some programming languages over others, just as different athletes prefer different kinds of equipment for their sport, but all programming languages are tools for the accomplishment of tasks other than to convey information to human beings. Consequently, computer programming languages, if they are to have intellectual property protection at all, must seek them under patent law rather than copyright law.[67]

In summary, the basic failure of many courts that have considered the scope of copyright protection in computer programs has been to ignore the crucial difference between patent and copyright subject matter. While Congress intended to eliminate that distinction to some extent by placing technological subject matter, in the form of computer program code, under copyright, Congress said nothing about its intended *scope* of protection.[68] In view of the nearly 200-year-old distinction between technological information and information whose sole appeal is to human understanding and esthetic appreciation, and given that Congress said nothing about the extent it intended copyright to intrude into the traditional realm of patent, it is prudent to assume that Congress intended as minimal an invasion as possible consistent with the known goal of protecting computer programs from piracy—verbatim copying.[69] That goal is achieved by limiting the scope of copyright protection for computer programs to literal program code and close "paraphrases" of program code.[70] This allows patent law, the traditional branch of intellectual property law that aims at promoting technology, to continue its work with so-called "nonliteral" elements of programs, such as program structure (or SSO).[71] Program interfaces, which are the product of programs and not programs in themselves, should be deemed protected by copyright only to the extent they stand independently as copyright subject matter, such as pictorial or graphic works.[72] And even where an interface product does meet the definition of copyright subject matter, it is subject to all of the traditional limitations on copyright protection, such as section 102(b).[73]

[66] Stern, *supra* note 63, at 370.

[67] *See* A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection for Computer Programs*, 72 NEB. L. REV. 351, 400–02 (1993).

[68] *See* Karjala, *supra* note 10, at 448–58.

[69] *Id.* at 446.

[70] Karjala, *A Coherent Theory*, *supra* note 14, at 73.

[71] *Id.* at 79–80.

[72] *Id.* at 94.

[73] *Id.* at 75–76, 99–100. Of course, courts have the power, and indeed the duty, to interpret statutes to make sense, which can sometimes mean deviating from the statutory language. Courts

VI. THE CASE LAW

Although *Altai* and *Lotus* have long been thought the leading authorities for the protection of computer program structure and program interfaces, respectively, a brief discussion of the notorious *Whelan* case is appropriate because of its resurrection by the Federal Circuit in *Oracle*. Here the plaintiff Whelan had written source code for managing a dental laboratory that defendant used as a starting point for writing source code to perform the same function on a different computer, with a different operating system and therefore different instruction set.[74] No evidence of literal code similarity was presented; indeed, the substantial similarity issue boiled down to a file structure and five specific subroutines.[75] The Third Circuit upheld the trial court's finding of infringement, holding that the program copyright covers the program's "structure, organization, and sequence."[76] The court recognized that technological and organizational efficiencies could make the program more or less valuable[77] but failed to inquire into the danger that the long-term of copyright protection might give longer protection to technological efficiency than even a patent would give, assuming that a patent were available for the particular technological advance. The *Whelan* court recognized that progress in computer technology was not qualitatively different from progress in other areas of science and the arts,[78] but it failed to recognize the *cumulative* nature of technological development. It bought into the "absence of merger" notion as a means of distinguishing idea from expression, even citing *Baker v. Selden* as authority.[79] However, *Baker* did not determine that there were no or only a few other ways of doing bookkeeping; it held that any *system* of bookkeeping had to look to patent law for intellectual property protection.[80] The *Whelan* court

do this, however, in the service of a policy goal that the literal statutory language does not achieve. In the case of computer programs, all of the policy is in favor of a *narrow* interpretation of the scope of protection, as discussed in the text: the statutory definition aligns itself closely with the policy goal of keeping separate realms of operation for copyright and patent.

[74] Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1225–26 (3d Cir. 1986).

[75] *Id.* at 1232–33.

[76] *Id.* at 1248; *see also* Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33, 76–80 (1987) (critiquing the *Whelan* decision).

[77] *Whelan*, 797 F.2d at 1230.

[78] *Id.* at 1238.

[79] *See id.* at 1234–36.

[80] Baker v. Selden, 101 U.S. 99, 102 (1879) (discussing the irrelevance of Selden's forms satisfying the minimal creativity requirements of copyright law as functional creativity is the province of patent law).

made other errors, but we need not belabor them except in the context in which the Federal Circuit in *Oracle* has resurrected the *Whelan* decision.

Computer Associates v. Altai[81] involved a microcosm of what the Java system tries to accomplish. Whereas Java seeks to apply "write once, run anywhere"[82] across a wide spectrum of computing devices, the programs at issue in *Altai* sought the goal of "write once, run on any one of three IBM operating systems."[83] The case is factually distinguishable from *Google v. Oracle*, however, in that it was the programs that did the actual conversions from the "write once" language into executable code for the different operating systems (analogous to the "Java Virtual Machine" in the Java system) that were allegedly infringed, not their interface (instruction set) as seen by the "write once" programmer.

The *Altai* court got off to a good start by recognizing that literal object and source code are protected by the program copyright,[84] meaning that its later filtration analysis does not apply to code. This is important, because if we were to filter even code for things like efficient operation, there would be nothing to protect at all (except perhaps, perversely, very badly written programs that, due to their inefficiency, would not be filtered out). The court, however, felt that it had to extend the program copyright to nonliteral elements, because that is what is done for novels and plays[85] (and failing to recognize that we do *not* extend to nonliteral elements for many other types of literary works). The *Altai* court sensed that protecting technological efficiency for the very long term of copyright could not be correct, so it conjured up, pretty much out of thin air but with the help of a few commentators,[86] an approach that divides the program into various "levels of abstraction" and, within each level, filters out all

[81] Computer Assoc. Int'l v. Altai, 982 F.2d 693 (2d Cir. 1992).

[82] Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1348 (2014).

[83] *Computer Assocs. Int'l*, 982 F.2d at 698–99.

[84] *Id.* at 702 (but failed to recognize that copyright protection does not extend to nonliteral elements for many other types of literary works).

[85] *Id.* at 702–03.

[86] David Nimmer, Richard L. Bernacchi & Gary N. Frischling, *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625 (1988); Steven R. Englund, Note, *Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866 (1990). The *Altai* court's primary criticism of *Whelan* was that *Whelan* allowed only one idea (the overall function performed by the program), treating the rest of the code as "expression." 982 F.2d at 705. This criticism of *Whelan* was correct as far as it went, but the division of the program into "levels of abstraction" had no basis in copyright doctrine. It in no way follows from Judge Learned Hand's "abstractions test," which draws a single abstractions line separating "idea" (more abstract) from "expression" (less abstract) elements of a traditional literary work. *See* Karjala, *A Coherent Theory*, *supra* note 14, at 79 n.75.

elements determined by efficiency considerations, dictated by external factors, or taken from the public domain.[87] The court failed to realize that *every* element of a computer program, literal or nonliteral, exists for the functional purpose of causing the computer to operate in a desired manner, subject to technological constraints that always compel tradeoffs in engineering designs.[88] Of course, some software engineers are better than others, so their designs might be deemed more elegant or even more creative, but these characterizations are never made in the abstract but rather with a view to how the program goes about executing its intended function. No one deliberately makes a program *less* efficient simply to achieve some purely esthetic objective in code that no one ever sees.[89] Therefore, an honest application of *Altai* necessarily leads to the correct policy-based conclusion that only the program code is protected (because the court had already stated that code was not subject to the filtering process).[90]

The *Altai* court was one of the few to recognize explicitly the crucial distinction in the statutory definition of "computer program" between the code—the set of statements or instructions—and the "certain result" brought about by execution of the code. The court expressly denied application of its abstraction-filtration-comparison analysis to the interfaces generated by the code:

> [W]e note that our decision here does not control infringement actions regarding categorically distinct works, such as certain types of screen displays. These items represent products of computer programs, rather than the programs themselves, and fall under the copyright rubric of audiovisual works. If a computer audiovisual display is copyrighted separately as an audiovisual work, apart from the literary work that generates it

[87] *Computer Assocs. Int'l*, 982 F.2d. at 706–10.

[88] Englund, *supra* note 86, at 869.

[89] Of course, writing "inefficient" code to achieve some esthetic *output* or result—whatever "inefficient" may mean in this context—is at least conceivable. Perhaps some particular result is only achievable by means of code that, in more normal contexts, would be considered "inefficient." Here we are considering whether any code writer would deliberately write code that achieves the desired result itself less efficiently than more efficient code that achieves the same result.

[90] *Computer Assocs., Int'l*, 982 F.2d at 710, 714 (suggesting that some "golden nugget" of protected expression might survive the filtering process, and even that some quantitatively small taking might be a "qualitatively vital aspect . . . of protected expression"); *see also* Karjala, *A Coherent Theory*, *supra* note 14, at 80 (stating that once elements related to efficiency and compatibility have been filtered out, however, it is difficult to see how anything remaining could even be important, let alone "vital").

> (i.e., the program), the display may be protectable regardless of the underlying program's copyright status.[91]

While the *Altai* court apparently did not have instruction sets and menu command hierarchies in mind, there is no principled difference between program code that generates a video game character and permits the user's moving him around in a designed environment and program code that generates an instruction set with associated syntax that permits a user—an application programmer—to write instructions using that instruction set to accomplish more specific results.

Unfortunately, the First Circuit in *Lotus v. Borland*,[92] failed to note the crucial statutory distinction between program code and the certain result achieved by execution of program code. Rather, the court thought that *Altai* had no application to the menu command hierarchy at issue because *Altai* involved nonliteral copying of code whereas *Lotus* involved literal copying of the Lotus 1-2-3 command structure.[93] Hanging up as it did on the question of whether the copying was "literal" or "nonliteral," the First Circuit failed to ask how the menu command hierarchy was copyright subject matter at all. As the *product* (or "certain result") of a computer program, the command structure is distinct from the code constituting the program. It therefore cannot be covered by the program copyright and must, if it is to be protected at all, be protected as an independently protectable work. As a result of this oversight, the First Circuit saw a potential for copyright protection as a part of the program copyright and only extricated itself from this needless predicament by finding (correctly) that the menu command hierarchy was a "method of operation" excluded from copyright protection by section 102(b).[94]

[91] *Computer Assocs., Int'l*, 982 F.2d at 703.

[92] Lotus Development Corp. v. Borland International, Inc., 49 F.3d 807 (1st Cir. 1995).

[93] *Id.* at 814–15.

[94] *Id.* at 815; *see also* Mitek Holdings, Inc. v. Arce Engineering Co., 89 F.3d 1548, 1555–57 & nn.15, 19 (11th Cir. 1996) (treating the user interface of a wood truss layout program as a nonliteral element of the computer program generating the interface but concluded, correctly, that it was an unprotectable "process" or "method of operation" under § 102(b); recognizing also, the role of patent law in protecting processes and methods).

It is of course the case that the program *code* is also a method of operation, but Congress's decision to include computer programs as copyright subject matter necessarily means, at a minimum, that the protection of program code is an implied exception to section 102(b). There is no evidence that Congress, in expressly deciding to protect easy-to-copy code under copyright, intended to protect any other technological features of programs or computers generally under copyright rather than their traditional protective home under patent law.

The First Circuit's basic problem, therefore, was the same as that for the *Altai* court, namely, separating protected from nonprotected elements ("idea" from "expression") by abstract reasoning and without reference to the role of patent law under our intellectual property scheme in protecting technological innovation. Many courts dealing with more traditional copyright subject matter have wrestled with the problem of the section 102(b) exclusion against a background of wholesale takings of factual information.[95] It may, of course, be correct that not everything we label in ordinary language as a "system" or other term appearing among the section 102(b) exclusions must automatically be excluded from copyright protection, but when these terms are in fact used as the best available descriptor of the aspect of the work in question, some explanation would seem to be in order for why a particular "system," for example, should nevertheless be protected by copyright in the face of section 102(b)'s denial of protection to "systems."[96] There is no need for this quandary for interfaces associated with computer programs, however, because both the policy for protecting code and the statutory definition of a computer program as program code—coupled with the absence of even a hint of congressional directive to the contrary—leave these technological features of program~~s~~ interfaces to their fate under patent or trade secret law unless they independently can be shown to qualify as copyright subject matter. While it is true that functional program interfaces, including programming languages and other means of operating a computer, are also excluded from copyright protection under section 102(b) even if they somehow qualify independently as copyright subject matter, it is rarely, if ever, necessary to go to that stage to eliminate them from copyright protection. Rather, as functional noncode aspects of computer programs, they simply are not copyright subject matter in the first place.

VII. APPLICATION TO THE FACTS OF *ORACLE V. GOOGLE*

This case deals with the copyright protectability of application program interfaces, or "APIs," which the Federal Circuit described as ready-to-use programs to perform specific functions on a computer and organized into

[95] *E.g.*, Warren Pub., Inc. v. Microdos Data Corp., 115 F.3d 1509 (11th Cir. 1997) (en banc) (refusing, over a strong dissent, to protect a creative selection of "principal communities" for listings of cable television services as a "system" under § 102(b)); Key Pubs., Inc. v. Chinatown Today Pub. Ent., 945 F.2d 509 (2d Cir. 1991) (protecting the "organizing principle" for a creative selection of categories for listings in a yellow-page directory aimed at the Chinese community, without recognizing that "principles" are among the § 102(b) exclusions).

[96] *See* Karjala, *supra* note 10, at 492 & n.217.

groups called "packages."[97] At the most basic level is code for performing a specific operation or function, which is called a "method."[98] A third party programmer writing in the Java language could take advantage of these methods—units of code—to perform various functions without writing independent code to perform them. "Classes" specify certain groups of methods (functions) along with the variables and other elements on which the methods operate.[99] Finally, the classes and certain related interfaces are grouped into "packages."[100] An analogy used by the district court and accepted by the Federal Circuit was the "collection of API packages is like a library, each package is like a bookshelf in the library, each class is like a book on the shelf, and each method is like a how-to chapter in a book."[101]

Ambiguous use of the term "code," unfortunately, appears to be at the root of the Federal Circuit's error, although Google apparently contributed to it.[102] According to the court, both parties to the dispute divided the packages into two types of source code: declaring code and implementing code:

> Declaring code is the expression that identifies the prewritten function and is sometimes referred to as the "declaration" or "header." . . . The expressions used by the programmer from the declaring code command the computer to execute the associated implementing code, which gives the computer step-by-step instructions for carrying out the declared function.[103]

[97] Oracle America, Inc. v. Google, Inc., 750 F.3d 1339, 1347–48 (Fed. Cir. 2014) (stating "[t]his copyright dispute involves 37 packages of computer source code. The parties have often referred to these groups of computer programs, individually or collectively, as 'application programming interfaces,' or API packages.").

[98] *Id.* at 1349.

[99] *Id.*

[100] *Id.*

[101] *Id.*

[102] *Id.* at 1351 (stating "[a]lthough it is undisputed that certain Android software contains copies of the 37 API packages' declaring code at issue, neither the district court nor the parties specify in which programs those copies appear"). This "declaring code," however, in order to be "contained" as a copy in a Google package would be there only in some binary representation that would allow the Google implementation program to recognize the specific declarations. For example, any word processing program would necessarily contain binary code for the letter "w" that would allow the program to recognize that the letter "w" had been typed and to place the letter at the appropriate spot. The letter "w" is not part of the implementing code for the word processing program except in this very limited sense.

[103] *Id.*

Google's system used the headings (called "declaring source code" by the court[104]) verbatim but wrote its own implementing code for all functions, with two minor exceptions.[105] The implementing code for each function (or method), whether in Java or Android, constitutes a computer program—a set of instructions to be used in a computer to achieve a certain result, namely, the function for which it is designed.[106] This implementing code is invisible to the Java programmer, who simply invokes it by means of the appropriate declaration.[107] The implementing code is not written in the Java programming language. Rather, the high-level Java source code language is converted into executable object code for a particular computer via a two-step process.[108] The Java source code is first converted by the implementing code into "byte-code," which is the input to a given computer's machine-specific Java virtual machine.[109] This byte-code is then converted by the Java virtual machine into electronic object code that will run on that particular machine.[110] Thus, the implementing code in both Java and Android accomplish the same function—converting Java program instructions into a form that can be "understood" by the particular machine on which it is to run.[111]

The declarations, or headings, however, only become part of a computer program when they are used together with other declarations, statements, or instructions as part of a computer program written in the Java language. Therefore, to refer to these individual headings as "code" is misleading, in that "code" conjures up notions of computer programs, as in "source code" or "object code." Individually, these headings are not computer programs at all; rather, they are individual instructions that can be used when writing computer programs in the Java or Android programming languages. If the individual declarations are to be protected by copyright, against verbatim copying or otherwise, they cannot be protected as "computer programs." Indeed, the implementing code that responds to each heading produces the result required by the given heading. The heading, thus, is part of the "certain result" produced by the implementing code (which is a computer program, as

[104] *Id.* at 1350–51.
[105] *Id.* at 1351.
[106] 17 U.S.C. § 101 (2012).
[107] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977 (Cal. 2012) (overturned on other grounds).
[108] *Id.*
[109] *Id.*
[110] *Id.*
[111] *Id.* at 978 (explaining that the Android system has its own "virtual machine," called "Dalvik"). The implementing code for a given function would therefore not give the same result as the byte-code produced by the Java system.

discussed above) because the implementing code is written in such a way that it recognizes and accepts the given declaration and executes the function demanded by that declaration.[112]

As an example closer to everyday life, consider a word processing program like Microsoft Word. The program code for Word is written in such a way that when the letter "w" is typed from the keyboard a symbol for "w" appears on the screen at the location of the cursor. Thus, accepting this keystroke, together with those keystrokes associated with many other symbols and allowing their on-screen manipulation, is part of the "certain result" brought about by execution of the Word program. Neither the letter "w" nor the entire set of symbols accepted by Word nor even an entire novel written using the Word program is protected by the copyright held by Microsoft in the Word program, although of course many works written using Word as a tool would be independently protected as literary works, in which the individual Word writers would own the copyright.

Google decided, apparently, that writers of apps for its Android system would want to use the functions contained in thirty-seven Java packages with which most such programmers were already familiar.[113] Google included additional declaration possibilities not available in Java, again presumably because Google thought its app writers would like to make use of such functionalities.[114] Any app written for Android, therefore, could be incompatible with a system that can accept only Java. This, of course, represents a threat to Java if the Android system should become so popular that increasing numbers of app programmers start writing solely for Android devices. This competition between two incompatible technological systems, however, is not one that copyright law is designed to mediate. Both Java and Android are programming languages, and as such they are technologies for using computers. Some, but not all, and certainly not simply the *names* of the functions performed, of the aspects of Java are likely patent protectable. To the extent Oracle has a patent in such aspects, Google must either license or invent around. Where a patent is not available for a technological feature, however, copyright cannot be used as a kind of consolation prize. To do so would afford a very long-term copyright without any examination of the level of technological advance. This would eviscerate the role of patent law as the primary source of intellectual property protection for this technology.

---

[112] Oracle Am., Inc. v. Goggle Inc., 750 F.3d 1339, 1349 (Fed. Cir. 2014).
[113] *Id.* at 1350.
[114] *Id.* at 1351.

The Federal Circuit begins its analysis by noting section 102(b) and *Baker v. Selden*, but it immediately turns to the classification of computer programs as "literary works" and the general rule that copyright protection extends to nonliteral elements of such works.[115] It then asserts that both SSO and the user interface are nonliteral elements of a program[116] with no reference to the statutory definition of a computer program, or the *Altai* case cited by the Federal Circuit, which provide that the user interface is the *result* of program operation, not an element of the program itself.[117] The court cites the 1990's Ninth Circuit decision in *Johnson Controls*[118] for the proposition that whether a particular nonliteral element is protected by the program copyright depends on whether the element in question qualifies as expression of an idea as opposed to an idea itself.[119] *Johnson Controls* does therefore suggest that at least some nonliteral structural elements of computer programs can be protected. *Johnson Controls*, however, was not only a pre-*Altai* case but also dealt with the issue of whether to reverse the grant of a preliminary injunction.[120] The district court had found a likelihood of success on the merits and, on that basis, presumed irreparable harm—a standard that the Ninth Circuit affirmed as correct.[121] Moreover, as noted by the district court in *Oracle v. Google*,[122] the *Johnson Controls* court supplied no standard for determining which program structures would qualify for copyright protection, other than noting that the programs at issue were complex so there appeared to be room for individualized expression.[123] In other words, "absence of merger" was the only test applied with no concern for the potential to protect technological efficiencies with copyright.[124]

---

[115] *Id.* at 1354–55.

[116] *Id.* at 1355–56.

[117] Computer Assocs. Int'l v. Altai, 982 F.2d 693, 703 (1992); *see also* 17 U.S.C. § 101.

[118] Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989). *Oracle v. Google* arose in the Northern District of California, but because the original action involved patent claims, appeal was to the Federal Circuit, which purported to apply Ninth Circuit law.

[119] Oracle Am., Inc. v. Google, Inc., 750 F.3d 1356 (Fed. Cir. 2014) (citing *Johnson Controls, Inc.*, 886 F.2d at 1175).

[120] *Johnson Controls, Inc.*, 886 F.2d at 1174.

[121] *Id.* (explaining that since that time, based on the Supreme Court's patent decision in a permanent injunction case, eBay Inc. v. MercExchange, L.L.C., 547 U.S. 388 (2006), courts have held that irreparable harm must be shown independently both for preliminary injunctions and in copyright cases). *E.g.*, Saliniger v. Colting, 607 F.3d 68 (2010).

[122] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 992 (2012).

[123] *Johnson Controls*, 886 F.2d at 1175–76.

[124] The *Johnson Controls* court cited the special master's report that "although it is common for process control software packages to include provisions for collecting historical data, and using various integration and averaging schemes to do so, it is unusual to implement this function as a point type, as Johnson did." *Id.* at 1176. Even if we assume the plaintiff was creative in

The Federal Circuit did recognize that the Ninth Circuit had endorsed the *Altai* test for separating protected from unprotected nonliteral elements of a computer program.[125] However, relying solely on cases dealing with traditional copyright subject matter rather than computer programs, the Federal Circuit treated the filtration step of the *Altai* procedure as raising issues only as to infringement, not copyright protectability.[126] On this basis, *everything* gets through the *Altai* filters, so neither the "declaring code" (i.e., the declarations or headings, which were the only things that were copied "literally") nor the SSO of the Java packages were precluded from copyright protection.[127] Thus,

implementing the given function as a "point type," no evidence was presented that this implementation was done for esthetic as opposed to functional reasons. If the implementation as point type was a distinct technological improvement over the prior art, the plaintiff would have been entitled to a patent on the improvement. If the improvement did not qualify for a patent—the intellectual property statute designed for the protection and promotion of technology—to give it copyright protection affords a much longer monopoly in the improvement with no showing that the improvement benefitted the public enough to qualify for a twenty-year patent.

[125] *Oracle Am., Inc.*, 750 F.3d at 1357.

[126] *Id.* at 1358. It is one thing, for example, to say that the language of an insurance contract is protected by copyright but only infringed by nearly verbatim copying. Continental Casualty Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958). That eliminates the need for courts to scrutinize contract language in the abstract and forces would-be drafters of contracts aimed at achieving the same result to redraft where possible, keeping terms of art and including provisions covering most if not all of the same items. It is quite another thing to say that a *technological* work that adopts a similar structure for reasons of compatibility or efficiency infringes, even if the adoption is nearly exact, when exact similarity is necessary for achieving the desired compatibility or efficiency. The Federal Circuit's approach essentially eliminates the crucial filtering step of *Altai*. In this case, it is safe to say that Google *did* rewrite as much of Java as it could while still remaining compatible, with respect to the thirty-seven functions in question, with third-party programs written using these thirty-seven functions. *Oracle Am., Inc.*, 750 F.3d at 1350–51. So, even if we apply merger analysis only at the infringement stage—that is, the copyright in the Java programs is conceded so the only issue is whether Google infringed—anyone comparing the two would immediately see that the vast amount of computer code actually doing functional work was written independently. The only thing left was the names of the programs, i.e., the "declarations." *Id.* at 1352. However, merger denies copyright in the declarations because the same names must be used to ensure compatibility. *Id.* at 1368. Google struggled with this aspect of its case, at times arguing that the merger doctrine denied protection to the declarations. *Id.* at 1370. There should ultimately be no copyright in the declarations at all because they are the result of computer program operations and do not themselves constitute a computer program. Assuming the declarations are somehow a part of the program copyright, though, merger decided "at the copyrightability stage" could have denied copyright to the entire set of programs. *Id.* at 1339, 1359. In *Continental Casualty*, the court denied protection to an entire insurance contract because most of its clauses were standard and could not be rewritten without fear of inviting a different interpretation to the terms in question. *Cont'l Cas. Co.*, 253 F2d at 702. There was no need for the Federal Circuit to get into any of this, because the declarations, as a bare set of headings, are not copyright subject matter. *Oracle Am., Inc.*, 750 F.3d at 1352.

[127] *Oracle Am., Inc.*, 750 F.3d at 1364.

verbatim copying of the "declaring code"[128] necessarily infringed, and that result did not change even assuming the same headers were necessary for compatibility with the abilities of third-party programmers who had become familiar with the Java system.[129]

This approach is entirely incompatible with the *Altai* filtration approach to determining *the copyright-protected elements* of a computer program. Rather than decide how high up the abstractions ladder courts need to go before finding "idea,"[130] the Federal Circuit simply treated all SSO elements that were not eliminated by merger analysis as copyright-protected,[131] leaving only infringement analysis in the comparison of the respective SSO elements of the two programs. But if an element of any work (not just a computer program) is copyright protected, verbatim copying of that element will always infringe (absent fair use). This approach thus abandons the *Altai* attempt to separate protected from unprotected elements of a computer program based on notions of efficiency and compatibility.[132]

We return to the compatibility question shortly, but first we must analyze the Federal Circuit's approach to the issue of the SSO of the Java packages. As discussed above, "creativity" alone is insufficient to bring an element of a work under the protection of the work's copyright, and neither does the availability of a number of ways to achieve a given function imply copyright protection for each specific way of doing it.[133] The Federal Circuit relied on the pre-*Altai Johnson Controls* case to conclude that the Ninth Circuit never adopted the "method of operation" reasoning of the First Circuit in *Lotus v. Borland*.[134] *Johnson Controls* did, indeed, state in dictum that computer program SSO is eligible for copyright protection, provided it was "expression" rather than "idea."[135] However, the Federal Circuit failed to understand the importance of *later* Ninth Circuit decisions expressly allowing (as fair use) the verbatim copying of an entire computer program for the purpose of learning and making

[128] *Id.* at 1359.

[129] *Id.* at 1371–72.

[130] *Id.* at 1356.

[131] *Id.* at 1367 ("[U]nder Ninth Circuit law, an original work—even one that serves a function—is entitled to copyright protection as long as the author had multiple ways to express the underlying idea.").

[132] Computer Assocs. Int'l v. Altai, 982 F.2d 693, 709 (2d Cir. 1992).

[133] See text accompanying *supra* notes 49–50, 52, 54.

[134] *Oracle Am., Inc.*, 750 F.3d at 1365–66.

[135] Johnson Controls, Inc. v. Phoenix Control Systems, 886 F.2d 1173, 1175–76 (9th Cir. 1989).

use of *unprotected* elements of such programs.[136] *Sony v. Connectix*,[137] in particular, should entirely have disposed of Oracle's claim to copyright protection for whatever creative SSO could be found in the grouping of its packages. In *Sony*, the entire program code was initially copied verbatim, so of course there was prima facie copyright infringement. The Federal Circuit dismissed *Sony* as a fair use case, which it was, but the use was fair because none of the copied code was used. The defendant had written independent code to reproduce the exact and entire command structure of the protected program (an operating system). To hold, as the court in *Sony* did, that the verbatim copying of the program code was a fair use because the defendant only made use of unprotected elements necessarily says that the command structure was *not* a protected element of the program. Java, too, is essentially an operating system, but one that works on all computers that have a compiler in the form of a Java Virtual Machine, regardless of their internal operating system. If the command structure and its organization was not protected in *Sony*, as the Ninth Circuit necessarily held, they cannot be protected elements of any of the program code constituting the Java system, either.

On interoperability, notwithstanding express Ninth Circuit language in both *Sega* and *Sony v. Connectix* that even verbatim copying of protected code was a fair use when necessary for achieving otherwise noninfringing compatibility with existing hardware or software, the Federal Circuit reached all the way back to another early case from the Third Circuit, *Apple v. Franklin*.[138] Infringement should have been easy and straightforward in *Apple*, because the defendant copied the entire Apple operating system verbatim. There was therefore no need to look into the scope-of-protection question, and the defendant's argument that copying was necessary in order to achieve compatibility with programs written by third parties (in which Apple, clearly, could claim no copyright interest) was easily dismissed in view of the evidence that Franklin had not even tried to write independent but compatible code. Unfortunately, the *Apple* court went well beyond what was necessary to decide the case, stating that the desire "to achieve total compatibility . . . is a commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged."[139] Given the technological

[136] Sony Computer Entertainment Inc. v. Connectix Corp., 203 F.3d 596, 601–03 (9th Cir. 2000); Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510, 1518, 1526 (9th Cir. 1992), *amended by* Order and Amended Opinion, D.C. No. CV-91-3871-BAC, Jan. 6, 1993.

[137] Sony Computer Entertainment, Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000).

[138] Apple Comp., Inc. v. Franklin Comp. Corp., 714 F.2d 1240 (3d Cir. 1983).

[139] 714 F.2d at 1253, *quoted in* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014).

nature of computer programs, compatibility is in fact a crucial consideration in determining the scope of protection, because if a competitor cannot build a compatible system (meaning one that presents the same instruction set and syntax), the program copyright covers not only the program code as intended by Congress but also gives a monopoly on the execution of programs written by *independent third parties*, because no competitor can create a compatible system.[140] By following the old *Apple* dictum instead of the more recent Ninth Circuit holdings, the Federal Circuit essentially denied that Google had any legitimate interest in trying to take advantage of the training and experience that many third party Java programmers had acquired.[141] This implies that only Oracle has a right to take advantage of the training hours put in independently and without pay from Oracle by the many third party Java programmers. The district court correctly recognized that Oracle's position on interoperability would give Oracle the exclusive right not simply to the version of program code in which Oracle held an undisputed copyright but to *all* program code that would implement the same set of functions to achieve the same result.[142] There is no indication that Congress intended copyright in computer programs to range so broadly. Indeed, it was the vulnerability of electronic *code* to quick and easy piracy—because patent protection is not available for most programs—that led to protection of code under copyright, notwithstanding its technological nature. Those seeking such an expanded application of copyright law to technology, where no evidence has emerged in over thirty years of steadily developing program technology that noncode aspects of program technology are similarly vulnerable, should have the burden of showing either that Congress intended such a result or, at a minimum, of giving a policy based reason for bringing copyright so much further into the traditional realm of patent.

By limiting its "copyrightability" analysis to creativity and absence of merger, the Federal Circuit failed to come to grips with computer programs as unique copyright subject matter. The approaches we take to determine the scope of protection in a traditional and nonfunctional literary work, such as a novel, cannot be blindly applied to functional technological works like computer programs. Everyone agrees that Congress intended to use copyright law to protect program code, and it necessarily follows that section 102(b) has been amended implicitly with respect to program code. The issue in this case is not

[140] *Sega Enterprises Ltd.*, 977 F.2d at 1523–24 (stating that in the context of promoting third-party competition via independently written code, the *Sega* court said, "an attempt to monopolize the market by making it impossible for others to compete runs counter to the statutory purpose of promoting creative expression").

[141] *Oracle Am., Inc.*, 750 F.3d at 1371.

[142] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 1001–02 (N.D. Cal. 2012).

program code, however, notwithstanding the Federal Circuit's repeated use of the term "code" in describing the copying that occurred. What was copied here was a part of the *product* of code—part of the "certain result" that is brought about by execution of the program (i.e., the code). Congress has not spoken on how far up the abstractions ladder copyright in the program code should extend. Nor has Congress given any hint that section 102(b) no longer applies to noncode but functional elements of computer programs such as SSO.

Program technology remains technology. Technology is protected under our intellectual property system with patent, and to some extent trade secret, law. Both patent and trade secret have internal limitations that are absent from copyright: in particular, patent lasts for only twenty years and must be passed by a qualified examiner as a nonobvious technological advance. Trade secret lasts only so long as the technology remains secret. To interpret copyright, which requires no examination for quality and lasts for ninety-five years, as protecting functional noncode aspects of programs allows copyright to trench deeply into the realm of patent and trade secret with no indication from Congress that such a disruption in the balances of our intellectual property system was desired or intended.

VIII. SUMMARY

Copyright protects the literal code of computer programs, but Congress has not spoken on the scope of that protection. Both policy and doctrinal analysis, however, show that protection in a computer program should be limited to literal code, and close paraphrases. Moreover, computer program interfaces are not protected by the program copyright because they are the product, the "certain result," of the operation of computer programs. Any aspect of a program interface that independently qualifies as copyright subject matter may be protected, but subject to all of the traditional limitations on copyright. The Java interface consists simply of a set of allowable instructions, each with a specific syntax, for use in building new computer programs. In other words, it is a programming language, essentially created and presented to the user (i.e., the Java programmer) by the Java Virtual Machine that comprises a part of nearly every computer now. The Java Virtual Machine is a set of complex computer programs and unquestionably copyright subject matter, but Google did not copy any of the code contained in the Java virtual Machine. It used only the "certain result" generated by execution of the Java Virtual Machine—

the available instructions and their syntax. In sum, the Java headings together constitute a programming language, which is not protected by copyright.[143]

[143] *See supra* note 67 and accompanying text.