Harvard Journal of Law & Technology
Volume 31, Special Issue Spring 2018

COMPUTER SOFTWARE AS COPYRIGHTABLE SUBJECT MATTER: *ORACLE V. GOOGLE*, LEGISLATIVE INTENT, AND THE SCOPE OF RIGHTS IN DIGITAL WORKS

*Ralph Oman**

TABLE OF CONTENTS

I. INTRODUCTION

It was not a foregone conclusion that Congress would choose to make computer software copyrightable. In the lengthy period of study preceding the 1976 Copyright Act, as well as afterward, some scholars advocated *sui generis* intellectual property protection for the digital code that would run certain hardware devices.[1] And even the commission that Congress convened to study the issue split in its recommendation.[2]

* Pravel Professorial Lecturer in Intellectual Property and Patent Law, George Washington University Law School. Former United States Register of Copyrights; Chief Counsel, Senate Subcommittee on Patents, Copyrights, and Trademarks of the U.S. Senate Judiciary Committee.

1. *See, e.g.*, Elmer Galbi, *Proposal for New Legislation to Protect Computer Programming*, 17 BULL. COPY. SOC'Y 280, 283–92 (1970) (proposing that the Copyright Act be amended specifically to add protection to computer programs); Joseph Scafetta, Jr., *Computer Software Protection: The Copyright Revision Bills and Alternatives*, 8 J. MARSHALL J. PRAC. & PROC. 381, 398–99 (1975) (proposing the creation of the "petty patent" with a shorter duration that would protect computer software based on elements of copyright law and patent law); Michael Alan Pope & Patrick Bruce Pope, *Protection of Proprietary Interests in Computer Software*, 30 ALA. L. REV. 527, 529 (1979) (proposing new legislation to protect computer software that would require both expressive and innovative ideas); Kay H. Pierce, *Copyright Protection for Computer Programs*, 30 COPY. L. SYMP. 1, 26–29 (1980) (proposing a "hybrid system" of protection where exclusive rights would be granted to the creator of a program for a shorter period of time than is granted to the recipient of a patent).

2. NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 1, 26, 27, 37 (1979) [hereinafter CONTU REPORT].

But when Congress made clear that computer programs would, in fact, be copyrightable, it effectively imported several centuries' worth of well-studied (if often misunderstood) legal doctrines to bear on the questions that would inevitably follow from its decision to categorize software as a "literary work," protectable like any other. Sorting out the proper application of these copyright doctrines — to operating systems, video displays, nested hierarchies embedded in functional menus, and other elements of software — has not always been straightforward.[3] But the essential, threshold proposition stands: by bringing software into the world of copyright, Congress plainly did not mean to *abrogate* longstanding copyright principles; it meant to subject software to them.

Are non-literal elements of computer software — for example, its "structure, sequence, and organization"[4] — protectable in their own right, separate and apart from the literal code? The answer is to be found in pre-existing principles of copyright protection: because the non-literal elements of "literary works" had, at the time, long been understood to be protectable, and because Congress chose to classify software as a "literary work," the non-literal elements of software are indeed protectable. There is no indication in the text or history of the Copyright Act to suggest otherwise. And it is no answer to say that computer software should be treated differently — with a thinner scope of protection in this and other respects — because it is "functional." Congress was well aware that computer software is inherently *functional* in respects that other literary works are not. Yet it still made computer software copyrightable.

It seems to me that Professor Menell, in his valuable contribution to this volume, implicitly embraces the view that Congress meant for the protection afforded computer software to be *different* from the protection afforded other works, because computer software is functional. From that premise, he argues that the Federal Circuit in *Oracle v. Google* erred at every turn. But the premise is, I think, mistaken. And to the extent that critics of *Oracle v. Google* base their complaints on the notion that the functional nature of software should

3. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l Corp., 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996) ("Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit.") (Boudin, J., concurring).

4. *See, e.g.*, Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989) (Computer programs are composed "of several different components . . . . Whether the non-literal components of a program, including the structure, sequence and organization and user interface, are protected depends on whether . . . the component in question qualifies as an expression of an idea, or an idea itself."); Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1350–51 (Fed. Cir. 2014) (defining Java's "structure, sequence, and organization" as "the elaborately organized taxonomy of all the names of methods, classes, interfaces, and packages — the 'overall system of organized names — covering 37 packages, with over six hundred classes, with over six thousand methods'").

yield a meaningfully narrower scope of protection from that afforded other works, it is incumbent on such critics to articulate precisely where Congress evinced that intention. As the Register of Copyrights in the era when many questions of first impression concerning the scope of digital rights initially presented themselves, I am not aware of any evidence that Congress ever did so.

II. Copyright Law And Computer Software

A. CONTU and the Protection of Software via Copyright

In 1980, as the world sat on the brink of the digital age, Congress amended the Copyright Act to provide computer programs the same copyright protection as all other literary works.[5] Its choice to do so, however, was far from preordained. Within academia and beyond, scholars and practitioners debated the proper type — and the proper scope — of protection for computer programs. Some recommended denying copyright protection to computer software altogether, observing, for example, that "[o]ne should become suspicious of the need for protection at present upon learning that the software industry is currently burgeoning without the use of copyright . . . ."[6] Others proposed protecting computer programs with specially tailored legislation, whether appended to an existing statute or in new, stand-alone provisions.[7] One scholar, for example, proposed creating what he termed "the petty patent," which would protect computer programs for a shorter time period than the typical seventeen-year patent, and which would require elements borrowed from both copyright and patent law: originality, novelty, and utility.[8] Another suggested that a special section be added to the Copyright Statute wherein Congress could account for the fact that "in some respects . . . a new computer program is similar to a new machine," but in other respects "a computer program can be duplicated with the same ease that one can duplicate a literary work."[9]

In view of the longstanding debate on the subject, Congress tasked a commission — the National Commission on New Technological Uses of Copyrighted Works ("CONTU") — with analyzing and recommending appropriate copyright protection for software. After careful study and consideration, CONTU released a final report in

5. Computer Software Copyright Act of 1980, Pub. L. No. 96–517, § 10, 94 Stat. 3015, 3018 (1980).

6. Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 344 (1970).

7. *See, e.g.*, Galbi, *supra* note 1, at 280–81; Scafetta, *supra* note 1, at 398.

8. Scafetta, *supra* note 1, at 398.

9. Galbi, *supra* note 1, at 281.

which it recommended that federal law treat computer software as copyrightable material.[10]

Congress adopted CONTU's recommendations wholesale, making its report particularly useful in terms of shedding light on Congress's intent.[11] In the report, CONTU made clear its view that software should be treated no different than any other work of authorship[12] — protectable if original. CONTU recommended that Congress amend the 1976 Copyright Act to "make explicit that computer programs, to the extent they embody an author's original creation, are proper subject matter of copyright."[13] Accordingly, under CONTU's recommendation, software would be protected as long as "the 'author' contributed something more than a 'merely trivial' variation, something recognizably 'his own.'"[14]

CONTU addressed at some length what protecting computer software should entail and how such protection would fit within existing copyright doctrines. Even then, it was apparent that the limitations contained in section 102(b) of the Copyright Act could pose particular challenges for computer software. That section states: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." Recognizing that essentially everything about software could, from a certain perspective, be characterized as falling under one or more of these rubrics, CONTU observed that "the distinction between copyrightable computer programs and uncopyrightable processes or methods of operations does not always seem to 'shimmer with clarity.'"[15] The majority of the commission thus thought it "important that the distinction between programs and processes be made clear."[16] To do so, it drew on the "venerable copyright case" *Baker v. Selden*, which it described as holding "that a valid copyright in a book describing a system of accounting, based upon the now-universal T-accounts, did not bar others from using that accounting system."[17] That holding, CONTU went on,

10. *See* CONTU REPORT at 12.

11. Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1519 n.5 (1992) ("Congress adopted all of the statutory changes recommended by CONTU verbatim. Subsequent Congresses, the courts, and commentators have regarded the CONTU Report as the authoritative guide to congressional intent.").

12. *See* CONTU REPORT at 12–13, 18–19.

13. *Id.* at 1.

14. *Id.* at 25.

15. *Id.* at 18.

16. *Id.* at 19.

17. *Id.* at 19 (citing Baker v. Selden, 101 U.S. 99 (1879) (discussing extensively the distinction between copyright law and patent law and finding that copyright does not give an author exclusive rights to "useful art" described in a book, only the description of such "useful art" is protectable by copyright)).

"is often misconstrued as imposing a limit on copyrightability of works which express ideas, systems, or processes." But, CONTU continued, "[a]s Professor Nimmer observes, 'the rationale for the doctrine of *Baker v. Selden* in no event justifies the denial of copyrightability to any work.'"[18]

Congress enshrined *Baker*'s holding in section 102(b) of the Copyright Act long before CONTU was formed. Under *Baker* and section 102(b), CONTU explained, "[c]opyright . . . protects the program so long as it remains fixed in a tangible medium of expression but does not protect the electro-mechanical functioning of a machine."[19] "Thus," CONTU concluded, "one is always free to make a machine perform any conceivable process (in the absence of a patent), but one is not free to take another's program."[20] CONTU's understanding accords with Congress's own description of 102(b) as "intend[ing] . . . to make clear that the expression adopted by the programmer is the copyrightable element in a computer program," but "that the actual processes or methods embodied in the program are not."[21]

Just as important as what Congress did do — protect computer software in the Copyright Act — is what Congress did not do. Congress rejected Commissioner John Hersey's view that computer software should not be protected by existing copyright law. Hersey voiced his vehement opposition to CONTU's recommendations in a dissent that consumed more than a quarter of CONTU's thirty-eight-page report. In it, he thoroughly addressed what he viewed as the many dangers posed by the Commission's recommendations. According to Hersey, "copyright is an inappropriate, as well as unnecessary, way of protecting usable forms of computer programs."[22] Warning against "distortion [of copyright] by shoehorn," Hersey continued:

> In the early stages of its development, the basic ideas and methods to be contained in a computer program are set down in written forms, and these will presumably be copyrightable with no change in the 1976 Act. But the program itself, in its mature and usable form, is a machine-control element, a mechanical device, which on constitutional grounds and for reasons of social policy ought not be copyrighted.[23]

18. *Id.* (quoting 1 MELVILLE NIMMER, NIMMER ON COPYRIGHT, § 37.31 (1976)).
19. *Id.* at 20.
20. *Id.*
21. H.R. Rep. No. 94–1476 (1976).
22. CONTU REPORT at 27.
23. *Id.*

In other words, there is — and always has been — copyright protection for the written expression of the initial plans of a program, but there should be no protection for the ultimate program because it merely controls a machine. Commissioner Karpatkin shared Commissioner Hersey's "doubts and concerns sufficiently to lead" her to dissent.[24] She also noted that "the late Commissioner Nix, who passed away before the Commission's final report, indicated that he shared them as well."[25]

Although Commissioner Melville Nimmer concurred in the Commission's ultimate recommendation, he too "share[d] in a number of the doubts and concerns expressed [by] Commissioner Hersey[] [in his] thoughtful dissenting opinion."[26] Nimmer was "most troubl[ed] [by] the Commission's recommendation of open-ended copyright protection for all computer software" because of the Commission's "failure to articulate any rationale which would not equally justify copyright protection for the tangible expression of any and all original ideas . . . ."[27]

Congress was unmoved.[28] Its decision to adopt the CONTU majority's recommendations, especially in the face of such criticism, makes it indisputably clear that computer programs are copyrightable in the same manner as all other works. As a result, "[t]he *sine qua non* of copyright" for computer software "is originality," a "requirement [that] is not particularly stringent."[29] As long as a work "possesses at least some minimal degree of creativity,"[30] then it merits copyright protection.

B. Functionality and Expressiveness Can Coexist

Perhaps the most important principle for purposes of understanding copyright protection for computer software is that a literary work can be both functional and expressive.[31] At the most basic level, all computer code is functional — it tells a machine, whether physical or virtual, what to do. And as the foregoing explains, CONTU recog-

24. *Id.* at 38.
25. *Id.*
26. *Id.* at 26.
27. *Id.*
28. Representative Robert W. Kastenmeier of Wisconsin, the legendary Chairman of the House Intellectual Property Subcommittee, stated in his floor remarks on final passage of H.R. 6933 that his legislation "eliminates confusion about the legal status of computer software by enacting the recommendations of [CONTU] . . . ." 126 CONG. REC. 29,895 (1980).
29. Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 345, 358 (1991).
30. *Id.* at 345.
31. Consider an instructional manual, for instance. Copyright typically protects the author's particular articulation of which steps to take to achieve the desired result — but does not, of course, wall off others from writing their own descriptions of how to achieve the same result.

nized that reality when recommending copyright protection for computer software and explained why it did not foreclose protection. When Congress acted on CONTU's recommendation, it demonstrated that it too understood the dual nature of a computer program, defining it as "a set of statements or instructions to be used directly or indirectly in a computer to bring about a certain result."[32] Even Professor Menell recognizes that "[c]omputer software" is "written work intended to serve utilitarian purpose . . . ."[33] So it cannot be that the functional nature of computer code precludes protection.[34]

Even before *Computer Associates Int'l v. Altai, Inc.*,[35] in which the Second Circuit examined and balanced the complexities of mixed functional and expressive aspects in software copyright, courts recognized that software should not be denied protection even if its expressive components are also functional. In *Apple Computer, Inc. v. Franklin Computer Corp.*,[36] for example, the Third Circuit considered Apple's claim that Franklin copied Apple's operating system in an effort to provide a competitive personal computer that also operated Apple programs. Franklin contended that operating systems were *per se* excluded from copyright protection under the express terms of section 102(b) because they are "put to utilitarian use."[37] The court rejected that view, instead distinguishing between "the method which instructs the computer to perform its operating functions" and "the instructions themselves."[38] The court observed that there is "nothing in the copyright statute to support the argument that the intended use or use in industry of an article eligible for copyright bars or invalidates its registration. We do not read such a limitation into the copyright law."[39] The court also noted that CONTU's majority rejected the expansive view some courts have given *Baker v. Selden*.[40]

32. 17 U.S.C. § 101 (2012).

33. Peter Menell, ***Error! Main Document Only.****Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. (SPECIAL ISSUE) 305, 315 (2018).

34. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1251 (3rd Cir. 1983) ("[t]hat the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability."); Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1372 (10th Cir. 1997) (holding that although an element may be "characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection. Section 102(b) does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation at a higher level of abstraction."); Apple Comput., Inc. v. Formula Int'l, Inc., 725 F.2d 521, 523–24 (9th Cir. 1984) (rejecting the notion that operating systems are "processes" and thus unprotectable).

35. 982 F.2d 693 (2d Cir. 1992).

36. 714 F.2d 1240.

37. *Id.* at 1252; *see also id.* at 1250.

38. *Id.* at 1251.

39. *Id.* at 1252 (quoting Mazer v. Stein, 347 U.S. 201, 218 (1954)).

40. *Id.* (citing CONTU REPORT at 21 ("[t]hat the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability."")).

The Third Circuit's *Franklin* decision does not stand alone. In *Toro Co. v. R & R Products Co.*,[41] for example, the Eighth Circuit explained that when a literary work, like computer software, is a system, the "particular expression" of that system is copyrightable even though, under section 102(b), the system itself is not. And in *Mitel, Inc. v. Iqtel, Inc.*,[42] the Tenth Circuit observed: "[A]lthough an element of a work may be characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection."[43]

While the utilitarian nature of software creates additional challenges, requiring courts to identify a program's protectable expression as distinct from the underlying function, denying copyright protection to software altogether is not the answer. Rather, existing methods such as the abstraction-filtration-comparison test, developed by the Second Circuit in *Altai*,[44] provide a roughly effective means for identifying when a follow-on software product infringes a pre-existing one, accounting for elements of the original that cannot properly supply the basis for copyright liability (*e.g.*, the use of *scenes-a-faire* and similar long-studied, well-understood elements of a work).

III. What Copyright Protection For Computer Software Means

Once Congress brought computer software under copyright's umbrella, questions abounded about how to apply traditional copyright principles to these relatively new works. The answers, almost invariably, lie in the Copyright Act and the doctrines that have developed around it. Although we now have almost forty years of experience with protection for computer software, it is still the source of much disagreement. When the basic principles of copyright law are understood, however, it becomes clear that the Federal Circuit in the *Oracle v. Google* case reached the right decision.

A. Defining and Protecting Software's Constituent Parts

Conspicuously absent from the Copyright Act is any limitation specific to the protection of computer software or its constituent parts. Nevertheless, identifying the line where protectable expression ends and unprotectable function begins requires defining both the work for which protection is sought, as well as the function it performs. Doing

41. 787 F.2d 1208, 1212 (8th Cir. 1986).
42. 124 F.3d 1366 (10th Cir. 1997).
43. *Id.* at 1372.
44. 982 F.2d 693.

so, however, can open the door to quite a bit of mischief, for in defining both concepts in a given instance, one can simultaneously remove wide swaths of software from copyright's orbit.

Professor Menell's analysis of the works at issue in *Oracle v. Google* provides a case study in the dangers posed by defining works at too granular a level. Take his assertion that "Oracle does not dispute that Google needed to include the particular declarations to make its Android platform perform the particular functions of the thirty-seven Java APIs."[45] Professor Menell seems to be implying that the functionality of the declarations and the expression of that functionality cannot be separated and thus the declarations are not protected. That view, in turn, is premised on his assertion that "the particular functions of the thirty-seven Java APIs" require the declarations.

In a limited sense, Menell's suggestion that the functionality of the declarations and their expression cannot be separated is of course correct: if the "function" to be achieved is framed as "the ability to write software code using the precise phraseology that the original author created," then it is in fact an *a priori* truth that there is no way to achieve *that* function except by including the same phraseology in the follow-on work. But the Copyright Act has never sanctioned such a tautological approach to defining the "function" a follow-on work is entitled to achieve in its own right, using its own creative expression. As the Federal Circuit explained, "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same *result*."[46] That is an eminently saner vision of the "function" for purposes of the analysis that Title 17 prescribes: look to what the thing does, rather than defining what the thing does such that its "function" necessarily encompasses its expression.

Professor Menell also says that the "API packages, unlike words, function as the gears and levers of a virtual machine."[47] He returns to this point again, stating:

> APIs function as the levers and gears of particular digital machines. The declarations must be reproduced to replicate the *particular* functionality. Android programmers needed to reproduce the same package (java.security), class name (ProtectionDomain), and method name (ClassLoader) to effectuate

45. Menell, *supra* note 33, at 443.

46. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014) (emphasis added). In fact, there were myriad ways Google could have organized the API packages and "nothing in the rules of the Java language . . . required that Google replicate the same groupings." *Id.* at 1352. Google thus had the ability to achieve the ultimate result it desired — utilizing the admittedly non-protected Java language.

47. Menell, *supra* note 33, at 364 n.316.

> a computer program that responds to the same inputs and produces the same outputs as the (java.security.machine).[48]

Here, too, it appears Professor Menell finds himself at least in the neighborhood of tautology. He seems to be suggesting that Google had to copy Oracle's packages — including both the declaring code and the SSO — because that was the only way to use Oracle's packages. Maybe so, but that does not mean using Oracle's packages is the only way to accomplish the desired function. Again, it is important to expressly acknowledge here that all of this turns to a large degree on the question: what is the function at issue?

The answer simply cannot be, "allowing users already familiar with the SSO of Oracle's packages to continue using them." If it were, then *any* follow-on work targeted at an existing base of adherents could indiscriminately replicate elements of the original that were necessary to the "function" of free-riding off of the pre-existing community's affinity for the creative, unconstrained choices made by the original author. It should go without saying that the Copyright Act simply does not tolerate that result.

Professor Menell further contends that "implementing code is the protectable computer program," but "declaring code constitutes 'the actual processes or methods embodied in the program [which] are not within the scope of the copyright law.'"[49] He asserts that this view is "faithful to the text and specific legislative history of [§ 102(b) of] the Copyright Act."[50] I respectfully disagree. There is no basis for drawing a distinction between types of code — let alone declaring and implementing code — for purposes of copyright protection.

Professor Menell's reliance on the Ninth Circuit's decisions in *Sega Enterprises Ltd. v. Accolade, Inc.*[51] and *Sony Computer Entertainment, Inc. v. Connectix Corp.*[52] also misses the mark. Contrary to Professor Menell's contention, neither case holds that "the software code necessary for interoperability is unprotectable by copyright law."[53] In fact, both cases were fair use cases in which copyrightability was addressed only tangentially. In *Sega*, the defendant made intermediate copies of Sega's software in order to reverse engineer it, to understand how Sega's game console interfaced with Sega's own game cartridges.[54] The *only* question before the court, however, was

48. *Id.* at 434.
49. *Id.* at 424.
50. *Id.*
51. 977 F.2d 1510 (9th Cir. 1992).
52. 203 F.3d 596 (9th Cir. 2000).
53. Menell, *supra* note 33, at 430.
54. *Sega*, 977 F.2d at 1514–15.

whether the defendant's intermediate copying was fair use; the Ninth Circuit was not presented with the question whether the software code that incorporated the functional aspects that the defendant sought to mimic also contained separable creative expression.[55] Nor was it presented with the question whether, notwithstanding the functional aspects of Sega's interface procedures, there was protectable expression in those procedures as well. Likewise, in *Sony*, the court addressed only whether the program at issue contained functional aspects, and it did so for purposes of determining fair use.[56] The court did not even consider whether Sony's software also had expressive aspects. [57]

Try as one might, one will find no place in the Copyright Act that distinguishes between the types of code in a computer program. "[T]he Act makes no distinction between the copyrightability of those programs which directly interact with the computer user and those which simply manage the computer system."[58] Nor does it make a distinction between code that invokes other components of a computer program — thus acting in a more "indirect" fashion — and code that implements aspects of a program on a machine. In fact, the Act specifically contemplates protection for both types of code in the definition of computer program. The definition of "computer program" establishes that copyrightability does not turn on how closely tied the computer code is to the ultimate function the computer performs.[59]

B. Non-Literal Similarity and Computer Software

Protection for literary works, in general, is not limited to the literal expression that the reader sees, i.e., the specific words on the page. Instead, copyright protection extends to the non-literal parts of a work, such as the contours of the plot of a play,[60] or the creative aggregation of pre-existing but previously separate design elements.[61] Thus, even if a follow-on work borrows none of the literal words of the original, it can still be infringing if it impermissibly copies non-literal elements that fall within the ambit of copyright protection.

In the context of software, in particular, "[i]t is well-established . . . that non-literal similarity of computer programs can constitute copyright infringement."[62] It is important to note that a giv-

55. *Id.* at 1514.

56. *Sony*, 203 F.3d at 598.

57. *Id.*

58. Apple Comput., Inc. v. Formula Int'l Inc., 725 F.2d 521, 524 (9th Cir. 1984).

59. 17 U.S.C. § 101 (2012) ("[A] set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.").

60. *See* Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930).

61. *See, e.g.*, Boisson v. Banion, 273 F.3d 262 (2d Cir. 2001).

62. Softel, Inc. v. Dragon Med. and Sci. Comm'ns, Inc., 118 F.3d 955, 963 (2d Cir. 1997).

en element of a computer program may be unprotectable in its own right, but that does not preclude protection for non-literal elements of the work as a whole, where they are assembled or organized in a creative fashion. As the Second Circuit has explained:

> In *Altai*, the district court held many aspects of the program at issue in that case to be not protectable for various reasons (e.g., because they were in the public domain or were computer *scenes a faire*). Nevertheless, the court proceeded to a higher level of abstraction and responded to the plaintiff's claim of infringement based on alleged similarities between the two programs' "organizational charts." . . . This Court approved that approach . . . .[63]

I have understood these principles to be non-controversial: of course the copyrightable elements of a complex work include particular combinations of features that reflect the author's creative choices, irrespective of whether the features themselves are independently copyrightable in their own right. And the implications of these principles for a software product like Oracle's in the *Oracle v. Google* case should be straightforward — though they have seemed to elude many commentators. If Oracle went out and created, from the ground up, a software program including 166 API packages; and if Google then went out and copied, down to the last jot and tittle, the precise structure of 37 of those packages; then Google engaged in a *prima facie* act of infringement of protected non-literal elements of Oracle's software program. We do not need to know anything else to reach that conclusion. It makes no difference whether, as Professor Menell insists, the declaring code in the API packages is independently copyrightable. If you create a cookbook consisting of 166 historic recipes, selected from the hundreds of thousands of recipes available — each of which is in the public domain and not subject to copyright protection in its own right — and I create a follow-on cookbook that copies the creative selection and arrangement of the first 37 recipes in your book, in precisely the same sequence and in deliberately exacting detail, then I have infringed the copyright in your cookbook, which you earned by selecting and arranging the recipes together in the first place.[64]

Compare Professor Menell's approach: he appears to believe that the non-literal elements of Oracle's software program — which he (like courts before him) calls its "SSO," for "structure, sequence and

63. *Id.* at 963–64.

64. More precisely, I have engaged in a *prima facie* act of infringement, which will yield liability if I have no defense — which I surely hope I do!

organization”[65] — are not copyrightable because the “declaring code constitutes ‘the actual methods embodied in the program [which] are not within the scope of the copyright law.’”[66] I understand him to mean that because the constitutive parts of Oracle’s program — the lines of code that he calls declaring code — are themselves uncopyrightable (in his view), then assembling them together in various hierarchies, sub-hierarchies, and sub-sub-hierarchies can never yield a protectable work (or part of one) that can be infringed by even studious replication. But that is simply not the law. Taking even *pre-existing* unprotected components and selecting or arranging them in a new way can result in a protected work that precludes copying of the original selection and/or arrangement.[67] *Creating* the components from the ground up, and then organizing them in complex hierarchies, is an even easier case.

Here, again, Congress has never suggested that these basic, longstanding principles of copyright law should cease to apply because the work at issue is a work of software. Perhaps it should have. Perhaps we would all be better off if, in the late 1970s and early 1980s, Congress had embraced the recommendations of those commentators who advocated *sui generis* protection for computer programs, and rejected the recommendations of those who preferred to bring computer programs into the universe of copyright law. I personally doubt that we would have done so, but I cannot rule out the possibility. As events in fact unfolded, however, *that is not the course that Congress chose*. And in view of this history, the burden should fall on those who advocate a departure from standard copyright principles for computer software to identify where, precisely, they understand Congress to have made such a radical departure part of our law. As a participant in some of the debates about copyright and software in the 1980s and ‘90s, I am hard-pressed to think of an instance when Congress did so (aside, of course, from computer-specific provisions of Title 17, which have nothing to do with the topics at issue here).[68]

IV. CONCLUSION

Despite Professor Menell’s ominous warnings about the import of the Federal Circuit’s ruling, the sky has not fallen. Nor has it fallen in the nearly forty years since Congress expanded the Copyright Act to protect computer programs — a critically important area of creative

65 Menell, supra note 33, at 326.

66. Id. at 424.

67. *See* Key Publ’ns, Inc. v. Chinatown Today Publ’g Enters., 945 F.2d 509, 513–14 (2d Cir. 1991).

68. *See, e.g.*, 17 U.S.C. § 119(a) (2012). Tellingly, Congress chose the *sui generis* option in 1984 when it enacted the Semiconductor Chip Protection Act.

development and innovation. Copyright protection continues to stimulate creativity, competition, and technological advancement. It suppresses piracy and predatory commercial practices. It encourages investment in new and better works. It contains the nuance necessary for complex technological environments. And, under the leadership of the United States, it has led to international consensus that computer programs are best protected with the application of general copyright principles. Diverting from this well-trod, proven path, chosen by Congress and relied on by innovators, requires more than policy arguments and disagreements with outcomes. The Federal Circuit rightly recognized as much and should be applauded for doing so.