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

THE REPORT OF API COPYRIGHT'S DEATH IS GREATLY EXAGGERATED

*Annette Hurst**

TABLE OF CONTENTS

I. INTRODUCTION

> A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.[1]

Let's properly frame the terms of the debate. The debate is *not* whether there should be protection of software functionality under copyright law. Nobody is advocating that software functionality be included as literal or nonliteral elements within the scope of copyright law. Google admitted that it could have written its own APIs to per-

* The author is a partner at Orrick, Herrington & Sutcliffe who has practiced copyright law in the United States for more than twenty-five years, and has participated in many high profile copyright cases including Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc., 964 F.2d 965 (9th Cir. 1992); Nintendo of Am., Inc. v. Lewis Galoob Toys, Inc., 16 F.3d 1032 (9th Cir. 1994); Mattel, Inc. v. Walking Mountain Prods., 353 F.3d 792 (9th Cir. 2003); Mattel, Inc. v. MGA Entm't, Inc., 616 F.3d 904 (9th Cir. 2010); Mattel, Inc. v. MGA Entm't, Inc., 782 F. Supp. 2d 911 (C.D. Cal. 2011); Kirtsaeng v. John Wiley & Sons, Inc., 568 U.S. 519 (2013); Fox Broadcasting Co., Inc. v. DISH Network LLC, 747 F.3d 1060 (9th Cir. 2014); and Oracle America, Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. 2014), as counsel for Oracle. She represents both plaintiffs and defendants in copyright infringement cases. The views expressed herein are her own and should not be attributed to Oracle, Orrick or any past or present clients. The author is grateful for the assistance of her colleagues Andrew Silverman, Nathan Shaffer, Matt Bush and Cheryl Watson for their assistance with this article.

1. 17 U.S.C. § 101.

form exactly the same functions as the Java APIs.[2] To the extent protected as intellectual property, functions are addressed in patent or trade secret law.

Instead, the debate is between, on the one hand, the categorical *exclusion* of expressive software elements from copyright protection because those expressive elements can also perform functions, and on the other hand, a fact-specific approach protecting original expressive aspects of software by separating idea from expression. The latter is what copyright law has always done — separate what is protected from what is not. 17 U.S.C. § 102 is worded to require such an approach — copyright protection *subsists* in expression under § 102(a) but the scope of its protection does not *extend* to methods under § 102(b). To say that the Federal Circuit's decision in *Oracle v. Google* was an unusual or even controversial extension of copyright principles exhibits a misguided understanding of the law and the facts, and ignores the statute and relevant legislative history.[3] Congress resolved the controversy over whether to include software expression within the ambit of copyright protection when it provided copyright protection for computer programs.

This comment will address two points in Professor Menell's article, both of which suggest there is no copyright to enforce in the relevant aspects of the Java APIs. The first is Professor Menell's suggestion that Sun was perfectly happy to have others copy the declaring code of the Java APIs without restriction, and that Oracle's enforcement of copyright in the Java APIs is a departure from Sun's and the community's views. The evidence showed the opposite. Sun obtained a copyright registration for the Java platform that included the Java APIs, separately licensed the declaring code of the Java APIs in its Specification License, enforced its copyright in the APIs by insisting that licenses be taken when they were used, and specifically lamented both internally and externally that Google's conduct with respect to Android was copyright infringement. Important members of the developer community likewise viewed Android's use of the Java APIs as copyright infringement.[4] There is no basis on which to conclude that Sun had somehow long ago ceded away the copyright question for the Java APIs or that Sun's actions were inconsistent with the outcome in the Federal Circuit's opinion.

The second point to be discussed below is Professor Menell's suggestion that software elements like the Java APIs' declaring code were categorically excluded from any copyright protection until *Ora-*

2. Peter S. Menell, *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, 411 (2018).

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

4. *See infra* Section III.

cle v. Google because of a legal consensus that they are a "machine."[5] Legal history shows otherwise. The National Commission on New Technological Uses of Copyrighted Works ("CONTU") recommended, and Congress adopted, copyright protection for the expressive elements of computer programs with full understanding that software is a machine.[6] The text of the statute says so.[7] CONTU's majority report expressly rejected the view that machines were categorically excluded from protection, and recommended defining computer programs to clearly acknowledge that software is a machine. Courts ever since have applied the same rules to software copyright as to other forms of literary works.[8] If software elements are expressive and not merged, then they are protectable. The Java API declaring code is no different. It is highly expressive, and Google abandoned any effort to prove that the expression merged into function. The Federal Circuit correctly decided the copyrightability question, and did so against a precedential backdrop that rendered this holding unremarkable.

In short, the Federal Circuit's opinion regarding copyrightability is no resurrection; rather, the report of the death of a copyright in the Java APIs is greatly exaggerated.

II. BACKGROUND AND TERMINOLOGY

The term API is used loosely to mean many different things, both in the Java context and in the larger context.[9] To make an assessment of the case, it is important not only to confirm a shared understanding of the relevant facts but also to clear away the fog of vague acronyms. First released in 1996, a distinguishing feature of the Java platform is the use of a virtual machine.[10] The virtual machine enables software

---

5. Menell, *supra* note 2, at 452.

6. CONTU stated that, "It was clearly the intent of Congress to include computer programs within the scope of copyrightable subject matter in the Act of 1976." National Commission on New Technological Uses of Copyrighted Works, Final Report Chapter 3 at 16 (1978) [hereinafter CONTU REPORT]; *see also id.* at 21 ("Nor has copyright been denied to works simply because of their utilitarian aspects. . . . That the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability."). Chapter 3 of the CONTU REPORT, concerning copyright protection for software, is available at http://digital-law-online.info/CONTU/PDF/Chapter3.pdf [https://perma.cc/D5RS-C7SA].

7. 17 U.S.C. § 101 ("A computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.").

8. CONTU REPORT, *supra* note 6, at 16; *see also*, *e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983); Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 838 (Fed. Cir. 1992).

9. Trial Tr. at 480:16–20 (May 11, 2016); 1217:22–1218:1 (May 16, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

10. Tim Lindholm, Frank Yellin, Gilad Bracha, & Alex Buckley, *The Java Virtual Machine Specification*, ORACLE 1, 2 (Feb. 13, 2015), https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf [https://perma.cc/3UQ5-D29M].

programmers to write programs able to run on different types of computer hardware without having to rewrite them for each different type.[11] A programmer could now write a program once and have the program work on any device, regardless of operating system.[12] "Write once, run anywhere," became the Java credo.[13]

Oracle v. Google is not about the Java programming language. It is not about the virtual machine.[14] It is not about a type of basic electronics communication protocols, sometimes also called interfaces, which enable computer devices to communicate with one another. Instead, the case is about computer programs that Sun and Oracle developers wrote using the Java programming language, which they called the Java Application Programming Interface, or "Java API."

Sun and Oracle wrote a vast array of computer programs and organized them into "packages" of source code.[15] The packages also contained further organizational subunits including, among other things, classes, interfaces and methods.[16] Each package consists of numerous modules of tried-and-true pre-packaged programs comprising a vast menu of functions.[17] Sun/Oracle's packages were a godsend to programmers who wrote apps for all sorts of devices. Instead of reinventing the wheel, all programmers had to do was write a few lines of code that called on those tried-and-true programs. The set of Java packages (and their elements) is referred to as the Java API, while one or more Java packages is referred to as a Java API or Java APIs, respectively.[18]

The basic concept is simple: every package consists of two related types of source code — declaring code and implementing code.[19] The declaring code is the line or lines of source code that introduce, name, and specify the package, class, or method.[20] It describes for an app programmer how to invoke or "call" a particular routine from the prewritten packages.[21] The declaring code embodies both literal and

---

11. *Id.*

12. *Id.*

13. *How Will Java Technology Change My Life?*, ORACLE, https://docs.oracle.com/javase/tutorial/getStarted/intro/changemylife.html [https://perma.cc/F5RU-8BBN].

14. Earlier in the case, patent claims were asserted that did concern the virtual machine, but those were not raised on appeal. This discussion is limited to the copyright issues.

15. *The Java Tutorials*, ORACLE, https://docs.oracle.com/javase/tutorial/java/concepts/index.html [https://perma.cc/B6RD-ETPS].

16. *Id.*

17. *Id.*

18. *Java Platform, Standard Edition & Java Development Kit Version 9 API Specification*, ORACLE, https://docs.oracle.com/javase/9/docs/api/overview-summary.html [https://perma.cc/DR87-7MQE].

19. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1349 (Fed. Cir. 2014).

20. Trial Tr. at 955:25–956:1 (May 13, 2012), Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 980-82 (N.D. Cal. 2012) (No. C 10-03561 WHA), *rev'd and remanded*, 750 F.3d 1339 (Fed. Cir. 2014).

21. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1349–50 (Fed. Cir. 2014).

nonliteral elements — it is both a compilable statement of code and it also comprises and defines the APIs' structure.[22] Implementing code is the source code that tells the computer how to carry out the declaring code.[23]

Writing any one of these packages is a creative and iterative process. It can take years. Much of the creativity lies in figuring out how to design a package and its elements — particularly all the declaring code — in a way that later programmers will find intuitive and memorable.[24] The process usually begins as an abstract high-level exercise. Developers identify a need in the Java community for new or different functions. Then, they organize a high-level summary of a possible structure for the package. For example, they wrestle with which functions to include in the package, which to put in other packages, and which to omit entirely, as well as how the various elements within the package relate to or interact with each other. They send sketches around to get comments from their colleagues, and may revise their design based on the feedback. The developers work with a clean slate, so ex ante, their options are infinite. Sun/Oracle invested hundreds of millions of dollars into this design process.[25]

In putting together Android, Google made verbatim copies of more than 11,000 lines of code comprising more than 7,000 class, interface and method declarations from 37 different packages of the Java Standard Edition.[26] Google acknowledged these were the most important packages for a mobile platform, and that it needed the Java API declarations to bring Android to market successfully and in time to compete in the burgeoning smartphone market.[27] When the structure of the relevant packages and classes is mapped, what Google took looks like this:

---

22. *Id.*

23. *Id.* at 1350.

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

25. Trial Tr. at 609:18–610:5 (May 11, 2016); 1352:11–13 (May 17, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

26. Menell, *supra* note 2, at *409.

27. Trial Tr. at 1121:5–13 (May 13, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); Brief for Google Inc. at 2, 15, Oracle Am., Inc. v. Google Inc., (Fed. Cir. May 22, 2017) (No. 17-1118), 2017 WL 2305681, at *2 (Google copied what was "key for mobile phones"); Trial Ex. 3211 at 61–62 (May 11, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938, (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) ("[I]f we are slow to develop products and technologies that are more compatible with non-PC communications devices we will fail to capture significant share of an increasingly important portion of the market for online services."); Trial Tr. at 633:2 (May 11, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (using Java was an "accelerant[]"); Trial Ex. 13, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); ("[A] shift to a primarily Java API" would "reduce our development time").

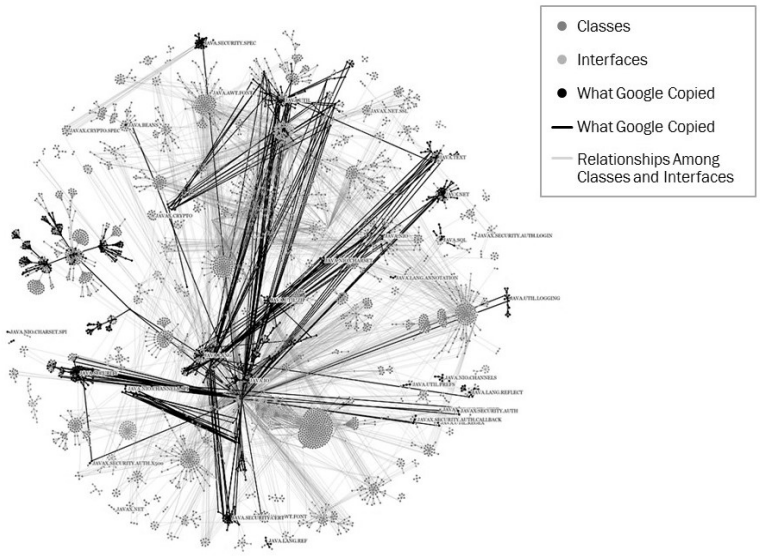

Figure 1: Pictorial representation of the structure of relevant packages and classes of the Java APIs

In short, Google copied both literal and nonliteral elements of the Java APIs when it copied 11,000 lines of declaring code and the structure and organization of 37 packages.

III. SUN'S LICENSING SCHEME AND VIEWS OF GOOGLE'S INFRINGEMENT

Professor Menell's article suggests, seemingly based on an earlier case involving Sun and Microsoft, that Sun did not believe it possessed a copyright in its Java APIs and that Oracle's enforcement was a departure from Sun's approach.[28] The publicly disclosed information in *Oracle v. Google* strongly indicates that Sun did not in fact hold this view or license in accordance with it. Indeed, a defense relying on this notion that Sun had already waived a copyright that Oracle later tried to enforce was expressly rejected after the first trial.[29]

Sun registered a copyright for the Java platform that included the Java APIs. As Professor Menell elsewhere acknowledges, Sun had (and Oracle now has) a multi-pronged licensing scheme for various

28. Menell, *supra* note 2, at 352–54, 375–78, 461.

29. Findings of Fact and Conclusions of Law on Equitable Defenses (May 31, 2012), Oracle Am., Inc. v. Google Inc., 872 F. Supp 2d 974 (N.D. Cal. 2012) (No. 10-CV-3561 WHA) (ECF No. 1203).

elements of the Java platform, including the API specification.[30] To accommodate all comers, Sun/Oracle offers three different licenses. One, the General Public License ("GPL") is free of charge, but subject to a strict and legally binding obligation[31]: licensees may use the packages (both declaring code and implementing code), but must "contribute back" the new work.[32] It is called an "open source" license, not because it is open for all to use unconditionally, but because the licensee must make his innovations publicly available. Two, the Specification License, unlike the GPL, does not permit the licensee to use the full Java source code.[33] Rather, the licensee can use only the Specification, which recites the declaring code.[34] So, a Specification licensee may write its own independent implementation of the Java APIs using the familiar declaring code and organization of the Sun/Oracle packages but must write its own implementing code.[35] Three, the Commercial License is for businesses that want to use and customize the full Java code in their commercial products and keep their code secret.[36] Oracle offers a Commercial License in return for royalties. Both the Specification and Commercial Licenses require that licensees' programs pass a series of tests (the TCK license) that ensure compatibility with the Java platform.[37] This compatibility requirement enforces adherence to Java's critical "write once, run anywhere" principle.[38]

---

30. Oracle did not seek to enforce any copyright that it may have had in the Java programming language. Menell, *supra* note 2, at 352–54, 357–359, 369; Trial Tr. at 1442–43 (May 17, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); Michael P. Doerr, Note, *Java: An Innovation in Software Development and a Dilemma in Copyright Law*, 7 J. INTELL. PROP. L. 127 (1999). During the second trial, Oracle stipulated that Google's use of those API declarations that were part of the Java language should be considered fair use. Order re 62 Classes and Interfaces (May 6, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1839); Trial Tr. at 458 (May 11, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

31. *See* Jacobsen v. Katzer, 535 F.3d 1373 (Fed. Cir. 2008).

32. Martin Lamonica, *Sun Picks GPL license for Java Code*, CNET (Feb. 14, 2007), https://www.cnet.com/news/sun-picks-gpl-license-for-java-code/ (last visited Jan. 27, 2018, 3:02PM).

33. *Java Specification License*, ORACLE, http://www.oracle.com/technetwork/java/javase/overview/javase8speclicense-2158700.html [https://perma.cc/LN3M-VFC6].

34. *Id.*

35. *Id.* The Specification license includes other terms designed to ensure both independent implementation and compatibility, chief among the latter is the requirement that a licensee fully implement and not "superset" or "subset" the API, as well as pass the Technology Compatibility Kit ("TCK") confirming compliance with the Specification. *Id.*

36. *Oracle Java SE and Oracle Java Embedded Products*, ORACLE, http://www.oracle.com/technetwork/java/javase/terms/products/index.html [https://perma.cc/P7ZA-P87Q].

37. *TCK Project Planning and Development Guide*, ORACLE 2, https://docs.oracle.com/javame/test-tools/jctt/tck_project_planning_guide.pdf [https://perma.cc/7P6S-WTDT].

38. *Id.*

Google thus had multiple license options for using the Java APIs in Android. It negotiated for a Commercial License, but ultimately refused all of Sun's proposals.[39] Then Google simply took a substantial part of the Java API declaring code without a license. In doing so Google evaded not only commercial royalty obligations but also the significant licensing restrictions on Oracle's licenses.

By avoiding the compatibility requirements, Google created a version of the Java APIs that was not interoperable with all prior existing versions — thus destroying the "write once, run anywhere" principle that had formed the foundation of the Java platform. And by refusing the GPL, Google also permitted its vendors, the cellphone manufacturers, to avoid the copyleft publication requirements. Google was deeply concerned that the viral nature of GPL would scare away the device manufacturers and thwart its ability to launch a new platform. Google also knew that Sun was working on or licensing its own versions of Java for smartphones, and the head of Android clearly viewed Sun as a competitor during this period.[40] Google thus benefited from evading both economic and non-economic license restrictions when it copied the Java APIs without a license.

Sun likewise viewed Google's conduct as a serious problem. While much has been made of a blog post on November 5, 2007, by then-Sun President Jonathan Schwartz "welcoming" Google to the Java community, there is strong evidence that Sun had previously enforced its API licensing terms and that internally it viewed Google as an infringer.[41] Properly viewed, Schwartz's "welcome" was an effort to make lemonade out of lemons.

In the early 2000s Sun had learned that Danger, an early smartphone developer (headed by the same Andy Rubin who later founded Android), had used the Java APIs in developing its mobile platform.[42] Sun insisted that Danger take a license and pay royalties even though it had only used the Java API declaring code and not the implementing code.[43] Danger was a highly successful early smartphone platform based upon the Java API, selling millions of copies in the T-Mobile Sidekick.[44] It was Rubin's experience of taking a license at Danger that in part led him to later confirm internally

---

39. Email from Andy Rubin to Bob Lee (Aug. 11, 2007), Trial Ex. 230, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

40. Trial Tr. at 845:19–846:1; 914:24 (May 12, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

41. Menell, *supra* note 2, at 344.

42. Trial Tr. at 887:23–24 (May 12, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

43. *Id.* at 888:24–1; Trial Tr. at 1625:8–1626:9 (May 18, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

44. Trial Tr. at 620:19–21 (May 11, 2016) Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

at Google in email that he understood the "java.lang apis are copyrighted."[45]

Likewise, Sun was *not* happy with what Google had done in Android. As the experience with Danger amply demonstrates, Sun knew that it had valuable and enforceable intellectual property rights, including copyrights. That was not the issue. The problem, as the documents at the time confirm, was that Sun did not — and could not — know or understand initially the scope of what Google was planning, both in terms of what aspect of the Java platform it was actually using and with respect to its ultimate intentions to comply with available license terms. Before Android's release and a couple of days after his blog post, Schwartz wrote, "I have no clue what they're up to — my sense is they're playing fast and loose with licensing terms, and they're going to start pissing people off."[46] A few days later on November 12, 2007, Schwartz wrote an email suggesting that Sun should get "the press to ask Google if their platform will be in compliance with the Java specification."[47] As the Specification License would have permitted Google to use the Java APIs in Android by meeting compatibility requirements, it was not yet clear that Google would be unable to claim the benefit of that license because it had failed to comply with the licensing terms. Sun's PR strategy seemed to pay off, as articles were published raising Sun's concern including quotations from Sun's executives.[48]

Later, even after the initial release of the Android SDK, Sun still was not sure of Google's intentions. Sun executive Vineet Gupta wrote an email to Schwartz indicating that Sun did not know "[i]f Google is still using Java in its mobile platform."[49] Gupta explained that he had "sent emails to Andy [Rubin] requesting a discussion around what they are planning, and if they need Java licensing . . . with no response."[50] Gupta also gave recommendations for different strategies based on whether Google would, or would not, eventually take a license. Specifically, he advised that, if Android used Java:

> a) then they have to come for a license with us, and will need to be compatible … b) [if] they will decide to go the non-compliant non-licensed route — then we will need to go deal with them or their handset

45. Trial Ex. 18, *Oracle v. Google* (No. C 10-03561 WHA).

46. Trial Ex. 2368, *Oracle v. Google* (No. C 10-03561 WHA).

47. Trial Ex. 1055, *Oracle v. Google* (No. C 10-03561 WHA).

48. Trial Ex. 1048, *Oracle v. Google* (No. C 10-03561 WHA) (Sun executive Rich Green: "We're reaching out to Google and assuming they'll be reaching out to us to ensure these platforms and APIs will be compatible so deployment on a wide variety of platforms will be possible."); *see also* Trial Ex. 9116, *Oracle v. Google* (No. C 10-03561 WHA).

49. Trial Ex. 565 at 2, *Oracle v. Google* (No. C 10-03561 WHA).

50. *Id.*

> vendor for IP issues . . . or c) [if Google] leverage[s] opensourced [sic] . . . [then] we will have to wait and see if they are following all the GPL rules.[51]

Despite his "welcome" blog post, Schwartz complained bitterly internally about Google, while Sun's financial fortunes continued to plummet. "The Google thing is really a pain. They are immune to copyright laws, good citizenship, they don't share."[52] "They also take Java for Android, without attribution or contribution. This is why I love scroogle."[53] Sun was experiencing the worst financial conditions in its history, bleeding personnel, and in no position to take on a major litigation battle against Google.

A year later, at the time of the Oracle acquisition, Sun continued to view Google's failure to take a license as a significant problem. In Schwartz's first e-mail to Oracle CEO Larry Ellison after the purchase, he outlined an agenda for discussion that included "the battles with Adobe Flash/Google Android."[54] In filings with the European Union seeking approval of the merger, Sun specifically opined that Android was "an unauthorized derivate work of Java SE,"[55] expressing the view held by Sun's legal department at the time.[56] Accordingly, Sun's former management, including former Sun CEO and cofounder Scott McNealy, fully supported Oracle bringing suit against Google.[57] Indeed, Google stipulated that Schwartz would not testify that Sun believed it had no legal claim.[58]

Nor was there consensus among the broader community of engineers that what Google had done was appropriate because there was no copyright on the Java APIs. As previously noted, Android head Andy Rubin had written emails declaring that "the java.lang apis are

---

51. *Id.*; *see also* Trial Ex. 538, *Oracle v. Google* (No. C 10-03561 WHA) (Gupta email to Android founder Andy Rubin) ("Several people at Sun are asking me about Google's plan in supporting Java on the announced Google Phone software stack. Would like to understand anything you could share with us."); Trial Ex. 5300, *Oracle v. Google* (No. C 10-03561 WHA) (Gupta explaining that Google had not licensed Java IP and "anyone shipping would be taking IP and other risks while dividing the Java ecosystem.").

52. Trial Ex. 563, *Oracle v. Google* (No. C 10-03561 WHA).

53. Trial Ex. 1056, *Oracle v. Google* (No. C 10-03561 WHA).

54. Trial Ex. 2362, *Oracle v. Google* (No. C 10-03561 WHA).

55. Trial Ex. 5295 at ¶ 70, *Oracle v. Google* (No. C 10-03561 WHA).

56. Earlier drafts of the EU filing by Sun's longtime chief in-house intellectual property lawyer had characterized Google as a "deliberate infringe[r]" and "recidivist bank robber." Testimony of Edward Screven, Trial Tr. at 1329:16–30:16; 1331:10–25 (May 17, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

57. *See* Amicus Brief of Scott McNealy and Brian Sutphin, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. 3:10-cv-03461-WHA).

58. Joint Stipulation and [Proposed] Order re Testimony of Jonathan Schwartz, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1725).

copyrighted" based on his prior experience at Danger.[59] It was not only Rubin who understood this from his industry experience, but other key third-party engineers also understood that the Java APIs were a copyright problem for Android.

The Apache Foundation was working towards an Apache-licensed open source implementation of the Java APIs called Apache Harmony.[60] The work at Apache proceeded based on the understanding that its Java SE API implementation would be compatible with the Java API Specification, pass the TCK, and therefore comply with the Specification License. Google used at least some of the Harmony project for its implementing code. The very same Apache Foundation developers who had created the Apache Harmony implementation believed that Google was ripping Sun off and speculated that Oracle would do something about it. "[W]e are, in fact, infringing on the [Sun] copyright if we distribute something that has not passed the TCK and *we know that.* This makes us *already* doing illegal things (in fact, Android using Harmony code is illegal as well)."[61] "What Is Oracle going to do about Android's ripping off some of (now) their IP and getting away with it?"[62]

Professor Menell apparently believes that Sun had a different view at an earlier time, seemingly based on his experience offering expert advice for Sun in the *Microsoft* case. In that case, Sun initially sought and obtained a preliminary injunction on grounds that included copyright infringement.[63] Sun did so, as Professor Menell has himself characterized it, for one of the same reasons Oracle sought to enforce here — to preserve the "write once, run anywhere" principle.[64] The copyright claims in *Sun v. Microsoft* encompassed Java language keyword extensions and use of the Java Native Interface, which is seemingly fully consistent with Oracle's position in *Oracle v. Google*. Only after the Ninth Circuit ruled that certain contractual requirements in the Sun/Microsoft commercial license agreement were cove-

---

59. Trial Ex. 18, *Oracle v. Google* (No. C 10-03561 WHA).

60. The Apache open source license is a more permissive license without the copyleft restrictions of GPL used by Sun when it selected an open source license. *See Apache License 2.0 and GPL Compatibility*, THE APACHE SOFTWARE FOUNDATION, https://apache.org/licenses/GPL-compatibility.html [https://perma.cc/L9UC-NQBN]; *What is Copyleft?*, FREE SOFTWARE FOUNDATION, https://www.gnu.org/licenses/copyleft.en.html [https://perma.cc/4PUS-VZTA]; *Comparison of Free and Open-Source Software Licenses*, WIKIPEDIA, https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses [https://perma.cc/K56Q-5DGN].

61. Trial Ex. 5046, *Oracle v. Google* (No. C 10-03561 WHA).

62. Trial Ex. 9201, *Oracle v. Google* (No. C 10-03561 WHA).

63. *See* Sun Microsystems, Inc. v. Microsoft Corp., 21 F. Supp. 2d 1109 (N.D. Cal. 1998).

64. *See* Peter Menell, *API Copyrightability Bleak House: Unraveling and Repairing the Oracle v. Google Jurisdictional Mess*, 31 BERK. TECH. L.J. 1515, 1536 n. 96 (2016) ("Sun sued Microsoft over its efforts to undermine the WORA principle").

nants rather than conditions did Sun drop its copyright claims and focus on its unfair competition claims.[65] That ruling clearly made it harder for Sun to enforce its copyrights, since Microsoft's apparent breaches of the contract no longer voided the license agreement but instead limited Sun to suing for a contract measure of damages. In any event, whatever Sun decided in that particular case, it is clear by the time of *Oracle v. Google* that Sun believed it had a copyright in the Java APIs, had previously sought to enforce it, and was deeply dissatisfied with Google's conduct.

Sun obtained a copyright registration and used a Specification License focused on licensing the declaring code. Sun enforced the copyright in the declaring code, requiring Danger to take a license when it had used Java declaring code in its popular smartphone platform. Sun's executives at all levels internally lamented Google's conduct and publicly called Google an infringer in filings to the EU. Important developers in the community likewise viewed Google's conduct with alarm. Sun was in no financial position to wage a litigation war against Google between late 2007 and early 2009, and was then preoccupied with completing an acquisition. When Oracle took over it tried to negotiate a license, and, when that failed, Oracle sued. Nothing about Oracle's approach was in any way inconsistent with Sun's approach.

In short, whether we look to Sun's or the community's views, the key players all understood there was a copyright on the Java APIs and that Google's copying without a license was an infringement. As shown below, there was a similar consensus in the legal community.

IV. THE JAVA APIS ARE HIGHLY EXPRESSIVE IN A MANNER LONG PROTECTED

Professor Menell next suggests that there was a legal consensus that the Java APIs declaring code was unprotected.[66] He further proposes that a line can be drawn between the Java APIs declaring code and implementing code, and that the former is unprotected and the latter is not because there is a legal understanding that the API is a "machine."[67] This approach has it exactly backwards. The declaring code is the *most* expressive part of the APIs and the implementing code is the most machine-like aspect of the APIs. In all events, there is no basis for a legal distinction under the text of the statute and the case law, as both types of code meet the definition for computer pro-

65. *See* Sun Microsystems, Inc. v. Microsoft Corp., 87 F. Supp. 2d 992, 993–94 (N.D. Cal. 2000).

66. Menell, *supra* note 2, at 451.

67. *Id.* at 465.

grams and are not subject to any categorical exclusion under § 102(b). Indeed, Google abandoned any defense demonstrating the merger of idea and expression in the structure and organization of the Java APIs,[68] and the Federal Circuit's decision as to copyrightability was both inevitable and correct.

The Federal Circuit concluded that Google's copying of both literal and nonliteral expressive elements constituted copyright infringement.[69] This was neither a surprising outcome nor should it be a controversial one if two basic principles of copyright are accepted. The first is that a copyright subsists in an "original" work,[70] and a work is "original" if "it possesses at least some minimal degree of creativity."[71] The second is that copyright protection covers computer programs under § 102(b) in exactly the same way as it does any other work.[72]

By definition, a computer program is functional.[73] That is what computer programs do: they bring about results using the machine logic of computers. A "machine," the term much used here by Professor Menell, is "an apparatus constructed to perform a task or for some other purposes."[74] Performing a task is no different than "bringing about a result."[75] People use computers to perform tasks, and they operate software to assist them in deploying the capabilities of those computers. *Every* computer program *is also a machine* according to the definition of computer programs supplied by Congress and the ordinary parlance used by Professor Menell.[76]

Nor does the statutory definition provide a basis for distinguishing one type of computer program from another. This is a broad definition. Code is code under the statute. If the code operates directly to bring about a result, it is protected. If it operates indirectly to bring about a result, it is protected. If it compiles and runs on the computer as part of an executable, it is clearly part of bringing about the result. All source code that compiles into a binary executable form is, by definition, both a computer program and a machine. The fact that code

---

68. Brief for Google Inc. at 47–51, Oracle Am., Inc. v. Google Inc., No. 17-1118 (Fed. Cir. 2017) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

69. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014).

70. 17 U.S.C. § 102(a).

71. Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 345 (1991).

72. *See, e.g.*, Hutchins v. Zoll Medical Corp., 492 F.3d 1377, 1385 (Fed. Cir. 2007); Computer Assocs., Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992); Johnson Controls, Inc. v. Phoenix Control Systems, Inc., 886 F.2d 1173, 1175–76 (9th Cir. 1989); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983).

73. 17 U.S.C. § 101.

74. The Oxford English Dictionary 156 (2d ed. 1989).

75. 17 U.S.C. § 101.

76. Menell, *supra* note 2, at 342.

is a "machine" tells us nothing about its legal status under the Copyright Act.

Moreover, in the parlance of the Java APIs, it is the declaring code that organizes the pre-programmed packages of functions so that developers can understand those functions without ever having to look at the implementing code.[77] The declaring code teaches the nature of the implementing code, including its relationship to other packages and classes of code.[78] The declaring code announces the structure and organization of the API packages so that developers do not have to learn the implementing code.[79] This type of teaching is the essence of an expressive purpose. It is the implementing code that *performs* the operations specified.[80] To the extent that either of these types of code is closer to a "machine," it is the implementing code, not the declaring code.

The effort to define the APIs as a "machine" seems to arise out of a desire to characterize them as a "method of operation,"[81] and therefore categorically exclude them from the scope of copyright protection. Certainly, § 102(b) is a limit on the scope of copyright protection in any particular work. But to read § 102(b) as a categorical exclusion of all works that have a dual nature — both expressive and functional — would be to read the protection for expressive aspects of such works out of the Copyright Act. This characterization fails to recognize a common principle of intellectual property law that material can have a dual nature — one aspect of which is protected and another not.[82]

Professor Menell's argument that the *Harry Potter* analogy fails to hold because that work of entertaining fiction is not at all functional is unpersuasive regarding the application of § 102(b).[83] Carried to its logical end, this approach to § 102(b) would categorically exclude all expression if it were factual just as well as if it were functional, because one cannot own facts under copyright law. Charts, maps, textbooks, and substantial aspects of other types of non-fiction works would lose copyright protection under this approach. These are all

77. Trial Tr. at 956:9–13 (May 13, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); Trial Tr. at 1216:18–19 (May 16, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

78. Trial Tr. 956:9–13; 956:18–20 (May 13, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

79. *Id.* at 956:5–7.

80. *Id.* at 956:20–23.

81. 17 U.S.C. § 102(b).

82. *Cf.* Varsity Brands, Inc. v. Star Athletica, LLC, 136 S. Ct. 1823 (2016) (affording protection for expressive aspects of utilitarian design); 17 U.S.C. § 113(b); Trademark Manual of Examining Procedure § 1213.05(c) (affording protection for double entendre marks where one meaning is not merely descriptive).

83. *See* Menell, *supra* note 2, at 387.

"functional" works because they convey information that is not protected by copyright. With this expansive and novel logic, copyright law would not protect any computer software at all. This result is exactly the opposite of what Congress intended when it expressly brought computer programs within the scope of § 102(a). All computer programs, by definition, serve as a method to operate a computer. Operating a computer is their statutorily defined function.

This notion of software exceptionalism must be wrong. Congress explicitly chose to protect computer programs under copyright law nearly forty years ago,[84] and they were properly regarded as protected even before that. In 1964, the Copyright Office announced that it would accept computer programs for registration,[85] and Congress clearly regarded computer programs as literary works at the time it adopted the 1976 Act.[86] CONTU similarly confirmed that understanding in its 1978 Report,[87] and any lingering doubt was removed by the adoption of the 1980 amendments explicitly defining computer programs and defining certain limitations on the scope of protection in § 101 and § 117.[88] Congress adopted verbatim the definition proposed by CONTU: "A computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[89] Congress adopted this CONTU definition despite the dissent by Commissioner Hersey, who urged against copyright protection on the ground that computer programs are "machine-control elements."[90] In other words, CONTU and Congress explicitly rejected the notion that computer programs should not be subject to copyright protection because they are usable as "machines."

The fact that a computer program is also a machine thus tells us nothing about its copyrightability. Being a "machine" is part of the basic definition. There is no basis to conclude that Congress intended to define computer programs as actionable statements or instructions that are copyrightable subject matter under § 101, with exclusive rights under § 106 limited by § 117, only to undercut those provisions entirely in § 102(b). No credible approach to statutory interpretation leads to such nullities. Indeed, the Supreme Court has consistently

---

84. Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3028.

85. Copyright Office Circular 31D (Jan. 1965).

86. H.R. Rep. No. 94-1476, 94th Cong. 2d Sess. 54 (1976) ("The term 'literary works' does not connote any criterion of literary merit or qualitative value: it includes catalogs, directories, and similar factual, reference, or instructional works and compilations of data. *It also includes computer data bases and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves*.") (emphasis added).

87. CONTU REPORT, *supra* note 6, at 16; *see also* Boorstyn, Copyright Law § 2.21 (1981).

88. Pub. L. No. 96-517, § 10, 94 Stat. 3028.

89. *Compare* 17 U.S.C. §101 *with* CONTU REPORT, *supra* note 6, at 12.

90. CONTU REPORT, *supra* note 6, at 27.

rejected an interpretation of copyright that denies *all* protection to copyrighted works that include functional as well as expressive aspects.[91]

Since the earliest cases, courts have consistently treated computer programs as subject to the usual set of rules for assessing the copyrightability of literary works under the Copyright Act.[92] The broad definition of literary work, including not only works expressed in words but also in "numbers, or other verbal or numerical symbols or indicia,"[93] was a natural fit for software. Whether expressed in numbers (object code) or text (source code), software could be assessed with reference to an established set of principles for the evaluation of the scope of protection in literary works.

These rules developed, primarily in the Second Circuit, over decades of cases involving plays, films, books and other texts.[94] In a seminal case concerning the play *Abie's Irish Rose*, Judge Learned Hand confirmed the protection of nonliteral elements such as plot, while identifying the difficulty of separating the protectable sufficiently expressive non-literal elements of a text from an unprotectable idea:

> Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his 'ideas,' to which, apart from their expression, his property is never extended. **Nobody has ever been able to fix that boundary, and nobody ever can.**[95]

Judge Hand had it exactly right: determining the boundary between idea and expression is highly context-sensitive. And, of course, development of the record for application of doctrines such as *scènes à faire* and merger is context-specific as well. To assess whether something is a stock element or highly constrained when it is created,

---

91. Mazer v. Stein, 347 U.S. 201, 219 (1954); *see also* Star Athletica, LLC v. Varsity Brands, 137 S. Ct. 1002 (2017).

92. *Compare* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 696 (2d Cir. 1992) *with* Nichols v. Universal Pictures Corp., 45 F.2d 119, 120 (2d Cir. 1930).

93. 17 U.S.C. § 101.

94. *See* Nichols, 45 F.2d 119.

95. *Id.* at 121 (emphasis added).

one must know its antecedents. In other words, there are important factual predicates to these legal inquiries.

It was hardly surprising when the Second Circuit applied its standard mode of textual analysis (abstraction-filtration-expression)[96] to determine that the scope of copyright protection is "highly fact specific" in *Computer Assocs. Int'l v. Altai*, *Inc.*[97] The case plainly did not hold that entire categories of works could or should be excluded from protection under the Copyright Act.

Likewise, *Sega Enters. Ltd. v. Accolade, Inc.*[98] did not purport to categorically resolve anything about § 102(b) or the scope of copyrightability in computer programs.[99] After disassembling the code, Accolade created a manual that contained "only functional descriptions of the interface requirements *and did not include any of Sega's*

96. Abstraction-filtration-expression is a standard mode of analysis used for assessing copyrightability and infringement of text under the Copyright Act. The analysis involves breaking down the underlying work into its constituent parts as reflected in the various ideas expressed therein, examining each part to sift out unprotected material such as scenes a faire or merged elements, and then comparing the remainder to the accused infringing work to determine if original elements were copied. *See* Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 361 (1991); Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 714–15 (2d Cir. 1992). In the case of computer programs, think of it as simply performing the standard analysis but having to do so in another language. The workflow is the same, but the process is more difficult because of an added complication of using a different language in assessing the substance at each step of the analysis.

97. 982 F.2d at 714–15.

98. 977 F.2d 1510 (9th Cir. 1992).

99. Google subsequently argued that the Federal Circuit did not follow Ninth Circuit precedent in *Oracle v. Google*, but the Ninth Circuit had decided at least by 1989 in *Johnson Controls* that nonliteral software elements are protected, and *Sega* did not modify that holding. Subverting the adversarial process, Google went to the Ninth Circuit in *Bikram Yoga* and filed a request, without notice to Oracle, asking the Court to disapprove of the Federal Circuit's decision in issuing its own. The court did not do so, and the decision in *Bikram Yoga* is fully consistent with *Oracle v. Google*. *See* Bikram's Yoga College of India, L.P. v. Evolation Yoga LLC, 803 F.3d 1032 (9th Cir. 2015). In *Bikram Yoga*, the copyrighted book at issue described the yoga sequence as purely functional — a system or method designed to "systematically work every part of the body, to give all internal organs, all the veins, all the ligaments, and all the muscles everything they need to maintain optimum health and maximum function." *Id.* at 1038. There was no creative spark nor expressive intent, as the Copyright Office has described, and thus a yoga sequence is not properly the subject matter of copyright; it is not choreography because "choreographic authorship is considered, for copyright purposes, to be the composition and arrangement of a related series of dance movements and patterns organized into a coherent and expressive whole." Registration of Claims to Copyright, 77 Fed. Reg. 37,605 (June 22, 2012). Nowhere does the court describe the yoga sequence as expressive in nature, contrary to Google's admissions in *Oracle v. Google* that the Java APIs are not only expressive but highly creative. Thus, unlike the Java API, the yoga sequence as a healing art was singularly functional rather than having a dual nature of both expression and function. The Ninth Circuit employed exactly the same approach to § 102(b) as the Federal Circuit did in *Oracle v. Google* when it held that the scope of copyright in the *Bikram Yoga* book did not extend to the yoga sequence itself: "section 102(b) is not a limitation on what kinds of expressive works may be protected by copyright," but "a limitation on how broadly copyright extends." *See Bikram Yoga*, 803 F.3d at 1038.

code."[100] By this description, the manual set forth the functions that the interface was required to perform, that is, the steps necessary to implement a method of unlocking the console so that it would play the cartridge.[101] The opinion tells us nothing more about the nature of the expression at issue. All the parties apparently agreed that Accolade did not replicate any of the code in its commercial product.[102] This is in stark contrast to *Oracle v. Google*, where the API declarations *are code* that Google copied verbatim into Android. We have no idea from the public record in *Sega* what the functional descriptions of the interface requirements looked like or how they might compare either to the complex structure and organization of the 37 packages or the 11,000 lines of code at issue in *Oracle v. Google*.

Treating *Sega* like a broad holding that excludes from the scope of copyright protection anything that might be termed an "interface" is a fool's errand. One could just as easily declare that all Hollywood "treatments" are excluded from protection as ideas while "outlines" are included as expression. A treatment might be five pages long or a three-sentence elevator pitch, and an outline might be one page or twenty-five. The terms mean nothing without reference to the actual substance. It is a strange approach to read a case's terminology-bound holding as completely independent of any understanding of what that terminology actually meant in the case. It is this kind of categorical approach that the Ninth Circuit expressly rejected in *Sega*, when it was concerned that announcing all reverse engineering as unfair would lead to overbroad protection for the functional aspects of software.[103] A declaration that all interface code is not protected by copyright because it is a method of operation would gut the statute in the other direction.

Altai and *Sega* were both decided in 1992 and fueled a debate between those who argued what the law *should* be and those who correctly identified what the law *was*. While the former continued to agitate for *sui generis* protection for software,[104] the latter continued to affirm what Congress had already done.[105] Those who argue today that *Oracle v. Google* is unusual read *Altai* and *Sega* as enacting a categorical exclusion, which was the same exact argument favored by the dissenters in CONTU and explicitly rejected by Congress in 1980.

---

100. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1515 (9th Cir. 1992) (emphasis added).

101. *Id.* at 1524 n.7.

102. *Id.* at 1515.

103. *Id.* at 1525.

104. *See* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994).

105. *See* Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559 (1994).

Indeed, after *Altai* and *Sega*, courts consistently rejected efforts at categorical exclusions of software elements from copyrightability, carefully evaluating the particular circumstances of each case.[106] This sensitive approach led to a wide variety of outcomes for different types of software elements. And, if there are clear takeaways from an equally divided Supreme Court in *Lotus v. Borland* (after a well-respected district judge went one way and the First Circuit split up and went another),[107] they are the difficulties implicated in the issue of the scope of software copyright, and the high degree of care with which a practitioner must counsel clients regarding those difficulties. As established in *Altai* and in numerous other decisions, the analysis for determining the degree to which copyright protection extends to computer programs applies exactly the same as it does for any other literary work. The workflow is the same, even if the language is different.

Thus, for elements characterized as "command names" or "command hierarchies," courts engage in context-specific, fact-bound analyses, reaching different decisions on whether the software at issue should be afforded protection based on the facts of that particular case.[108] For elements characterized as input or output formats or inter-

106. *See infra*, text accompanying notes 108–09.

107. 516 U.S. 233 (1996).

108. *Compare* Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc., 307 F.3d 197 (3d Cir. 2002) (holding that use of copy and call commands from competitor's software was copyright infringement); Softel, Inc. v. Dragon Med. & Sci. Commc'ns, Inc., 118 F.3d 955, 966–67 (2d Cir. 1997) (holding that non-literal elements, including a hierarchy of menus, may be copyrightable and remanding for district court to apply filtration test); Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1348 (5th Cir. 1994) (recognizing that menu commands/hierarchy may be entitled to copyright protection subject to filtration analysis and reversing district court holding to the contrary); Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 843–44 (10th Cir. 1993) (remanding for district court to conduct filtration analysis on computer program menus to determine copyrightability); Cisco Sys., Inc. v. Huawei Techs., Co., 266 F. Supp. 2d 551, 554 (E.D. Tex. 2003) (explaining that under Fifth Circuit law, a command hierarchy (including names) may be copyrightable as a non-literal element*);* Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202, 211 (D. Mass. 1993) (holding that Lotus 1-2-3 "menu tree" is copyrightable), rev'd, 49 F.3d 807 (1st Cir. 1995); Consul Tec, Inc. v. Interface Sys., Inc., No. 90-CV-70757-DT, 1991 WL 427891, at *1 (E.D. Mich. Oct. 31, 1991) (enjoining software that copied "commands, command phrases, and other aspects of the user interface" of competitor's software); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 68 (D. Mass. 1990) (holding that the menu/command structure of Lotus 1-2-3 is copyrightable); Digital Commc'ns Assocs., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 461 (N.D. Ga. 1987) (holding that arrangement and presentation of command names on a status screen was copyrightable) *with* Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1372 (10th Cir. 1997) (affirming non-copyrightability of codes for call control hardware that were created arbitrarily with no creative expression); MiTek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1556–57 (11th Cir. 1996) (holding that a menu structure that simulates the process by which roof plane trusses were hand drafted and permitted no deviation was not copyrightable, but reserving question as to whether any command structure could be copyrighted); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–16 (1st Cir. 1995); Real View, LLC v. 20-20 Techs., Inc., 683 F. Supp. 2d 147, 158–59 (D. Mass. 2010) (menu hierarchy of commands repre-

faces, results were mixed.[109] Data structures achieved more uniform protection.[110] File formats also experienced mixed outcomes.[111]

Nor did academics hold a uniform view that the 1992 decisions in *Altai* and *Sega* had effected a categorical exclusion of nonliteral software elements from the scope of copyright under § 102(b).[112] Articles published in the years following *Lotus v. Borland* reiterated the com-

---

sented by graphical icons not copyrightable); Wireless TV Studios, Inc. v. Digital Dispatch Sys., Inc., No. 07 CV 5103 (RJD) (RER), 2008 WL 2474626, at *2 (E.D.N.Y. June 19, 2008) (menu commands are uncopyrightable methods of operation); Jamison Bus. Sys., Inc. v. Unique Software Support Corp., No. CV 02–4887(ETB), 2005 WL 1262095, at *12–13 (E.D.N.Y. May 26, 2005) (menu command hierarchies are uncopyrightable); ILOG, Inc. v. Bell Logic, LLC, 181 F. Supp. 2d 3, 9–10 (D. Mass. 2002) (holding that "rules editors," commands through which software was operated, were not protectable as methods of operation); Mitel, Inc. v. Iqtel, Inc., 896 F. Supp. 1050, 1055 (D. Colo. 1995) (command codes for computer call controller not protectable expression).

109. *Compare* Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc., 307 F.3d 197, 219 (3d Cir. 2002) (holding that use of copy and call commands from competitor's software was copyright infringement); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1547 (11th Cir. 1996) (rejecting argument that interface specifications are not copyrightable as a matter of law); Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1346 (5th Cir. 1994) (holding that input formats may be eligible for copyright protection if they contain sufficient creativity and originality); SAS Inst. Inc. v. World Programming Ltd., No. 5:10-CV-25-FL, 2012 WL 5844910, at *7 (E.D.N.C. Oct. 18, 2012) (denying motion to dismiss copyright claims based on infringement of operations, syntax, and command structure of software); MedioStream, Inc. v. Microsoft Corp., 749 F. Supp. 2d 507, 520–21 (E.D. Tex. 2010) (denying motion to dismiss copyright infringement claims based on unauthorized use of API); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (finding that structure of program, including input formats, is potentially protectable) *with* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992) (in fair use factor two context "interface procedures" not entitled to broad scope of protection); Synercom Tech., Inc. v. Univ. Computing Co., 462 F. Supp. 1003, 1014 (N.D. Tex. 1978); Torah Soft Ltd. v. Drosnin, 136 F. Supp. 2d 276, 292 (S.D.N.Y. 2001) (output of software in particular format not protectable expression).

110. *See* eScholar, LLC v. Otis Educ. Sys., Inc., No. 04 Civ. 4051 (SCR), 2005 WL 2977569, at *20–21 (S.D.N.Y. Nov. 3, 2005) (holding that data structure created by computer program may be copyrightable); Positive Software Solutions, Inc. v. New Century Mortgage Corp., 259 F. Supp. 2d 531, 535 (N.D. Tex. 2003) (granting injunction against use of SQL data structures); O.P. Solutions, Inc. v. Intellectual Prop. Network, Ltd., No. 96 Civ. 7952(LAP), 1999 WL 47191, at *20 (S.D.N.Y. Feb. 2, 1999) (arrangement of data tables protected); CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 355 (M.D. Ga. 1992) (data structures protected). *See also* Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1242 (3d Cir. 1986) (file and data structures protected).

111. *Compare* SAS Inst. Inc. v. World Programming Ltd., No. 5:10-CV-25-FL, 2012 WL 5844910, at *6–7 (E.D.N.C. Oct. 18, 2012) (holding that complaint alleging ability of defendant's software to use file format developed by plaintiff stated claim for copyright infringement); Harbor Software, Inc. v. Applied Sys., Inc., 925 F. Supp. 1042, 1049–50 (S.D.N.Y. 1996) (format of database files); Control Data Sys., Inc. v. Infoware, Inc., 903 F. Supp. 1316, 1322–23 (D. Minn. 1995) (holding that file layouts are protectable expression); CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 355 (M.D. Ga. 1992) (holding that file formats are expression, not idea) *with* Prof'l Mkt. Research, Inc. v. AC Nielsen Corp., No. 03–2314 (GAG), 2009 WL 1259956, at *2 (D. P.R. May 4, 2009) (data tables in software uncopyrightable method of operations); Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1088–89 (D. Mass. 1996) (file structures not protectable expression).

112. *See, e.g.*, Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU*, 106 HARV. L. REV. 977 (1993) (criticizing *Sega*).

plexity of the question and could hardly be understood as an academic consensus regarding categorical exclusions from copyright. In his seminal work the following year, Professor Miller criticized *Sega* for allowing too much copying under the fair use doctrine,[113] and he preferred *Whelan's* broader scope of copyright protection to that of *Altai,*[114] but nonetheless confirmed the straightforward reading that *Altai* still afforded protection for nonliteral elements.[115] Neatly presaging the record in *Oracle v. Google*, Miller reaffirmed his view that there should be strong protection for interfaces, arguing that "user interface design is extremely resource-intensive, and a well-composed user interface is frequently the precise feature that renders a program successful."[116] Other commentators continued to note the dual nature of software as both expressive and functional as well as a strong desire to protect the expressive aspects of programs.[117] Still others argued that *Altai* was more "restrictive" than *Whelan*, but criticized *Whelan* because it had failed to follow the traditional patterns of abstraction analysis announced in *Nichols* — thus affirming that under *Altai*, nonliteral elements remained protected as they always had.[118]

In November 1993, industry and academia came together for the Symposium on Copyright Protection and Reverse Engineering of Software at the University of Dayton School of Law.[119] The articles that resulted took a variety of positions, but none argued that § 102(b) took away copyright protection for nonliteral elements or interfaces.[120] Professor Karjala thought it remained an open question whether interfaces were or should be protected.[121] Then came the Columbia symposium in 1994 with Professor Samuelson's *Manifesto* arguing for *sui generis* protection for software and Professor Ginsburg's *Par-*

113. *Id.* at 1013–26.

114. Whelan Assocs., Inc. v. Jaslow Dental Lab, Inc., 797 F.2d 1222 (3d Cir. 1986); *see* Miller, *supra* note 112, at 1001–11 (comparing *Altai* and *Whelan*).

115. Miller, *supra* note 112, at 1007–08 (noting *Altai's* protection for "non-literal program structure").

116. *Id.* at 1033–34.

117. John M. Greim Jr., Note, *Against a Sui Generis System of Intellectual Property for Computer Software*, 22 HOFSTRA L. REV. 145 (1993).

118. W. H. Baird Garrett, Note, *Toward a Restrictive View of Copyright Protection for Nonliteral Elements of Computer Programs: Recent Developments in the Federal Courts*, 79 VA. L. REV. 2091 (1993).

119. Robert A. Kreiss, *Introduction to Symposium: Copyright Protection and Reverse Engineering of Software*, 19 U. DAYTON L. REV. 837 (1994).

120. *See, e.g.*, Anthony L. Clapes, *Confessions of an Amicus Curiae: Technophobia, Law, and Creativity in the Digital Arts*, 19 U. DAYTON L. REV. 903, 974 (1994) (Assistant General Counsel of IBM criticizing industry amicus briefs as creating industry "whiplash" and preferring the "orderly accretion of precedent"); Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 U. DAYTON L. REV. 975, 990–91 (1994) ("The real question for user interfaces, therefore, is whether functionality at the user level — including user lock-in and standardization — is or should be copyright protected" arguing that the "two most recent authorities are in hopeless conflict").

121. Karjala, *supra* note 120.

adox rejoinder arguing that traditional copyright principles were more than up to the task to properly delineate the scope of protection in software.[122]

In 1995, Professor Lemley weighed in with his view that the debate had been resolved in favor of *Altai* and against *Whelan* as a method of conducting the copyrightability analysis in software, but he also confirmed that *Whelan*'s holding for protection of nonliteral elements in software was "uncontroversial."[123] Far from declaring anything dead, Professor Lemley notes that it was the expansive view of the scope of copyright for computer programs that had become "institutionalized."[124]

The First Circuit's decision in *Lotus v. Borland* came in 1995,[125] and the dean of the copyright bar, Professor Nimmer (the elder) made his pronouncement on the subject:[126]

> To comply with the congressional decision to protect computer programs as literary works, the answer must be that in some circumstances and to some degree the exclusion created for processes and methods of operation must be interpreted to permit copyright protection of code, structure, or commands even though they are essential to duplicate the operation of a particular program.

Patry, then at Cardozo (now in-house at Google), strongly criticized the First Circuit's opinion, calling it a "unique, grotesque understanding of how Section 102(b) works."[127] His overall take? That the case "involves nonmerged material copied in its entirety for purely commercial purposes in an attempt to deprive the copyright owner of its

---

122. *See* Ginsburg, *supra* note 105.

123. Mark Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 9–10 (1995) ("During the same period, a number of district courts took an approach similar to Whelan, at least to the extent of rejecting Synercom and protecting software against non-literal copyright infringement. On that issue, Whelan remains uncontroversial to this day. Indeed, the Fifth Circuit has since reversed Synercom and affirmed the copyrightability of non-literal elements of computer programs").

124. *Id.* at n.142 ("That this expanded view of copyright law in the software context has become institutionalized is amply demonstrated by the arguments of certain copyright scholars, who decry the recent trend away from broad copyright protection on the grounds that it will fail to reward programming efficiency") (citations omitted).

125. 49 F.3d 807, 815–16 (1st Cir. 1995).

126. Raymond T. Nimmer, *Sliding Scales and Abstracted Expression*, 32 HOUS. L. REV. 317, 338 (1995).

127. William F. Patry, *Copyright and Computer Programs: It's All in the Definition*, 14 CARDOZO ARTS & ENT. L. J. 1, 58–59 (1996).

market share."[128] That version of Patry would have been on Oracle's side in this case.

The Court divided in its review in Lotus in early 1996.[129] Does it need to be said that an equally divided Court cannot properly be viewed as a product of widespread legal consensus for the exclusion of nonliteral elements from the scope of software copyright? Apparently so.[130] Meanwhile, others began responding to Professor Samuelson's *Manifesto*.

Professor Gorman noted the importance of the international consensus for copyright protection of software in rejecting the proposed *sui generis* approach:

> The authors challenge the legal treatment of program code as literary text. Instead, they view programs as 'virtual machines,' and as the medium of creation (like steel or plastic) rather than the artifact; interfaces are said to be like gear teeth or pulleys in a physical machine. This view was squarely rejected in the United States nearly twenty years ago by [CONTU], and by Congress itself when in 1980 it expressly defined a computer program and declared it be a species of 'literary work' covered by the Copyright Act. In the intervening years, essentially all nations have come to treat programs in that manner, and recently, 124 nations subscribed to the [TRIPs] provisions of [GATT], which also treat programs not as machine parts but as literary works.[131]

Commentary published just after the Court divided in *Lotus v. Borland* reiterated the complexity of the question and certainly did not amount to a consensus regarding categorical exclusions of nonliteral elements from copyright.[132] In 1998 Professor Karjala again called it a

---

128. *Id.* at 63 n.273 ("For these reasons, Judge Keeton rightly rejected Borland's fair use defense").

129. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233, 233 (1996).

130. *See* Jonathan E. Retsky, *Computer Software Protection in 1996: A Practitioner's Nightmare*, 29 J. MARSHALL L. REV. 853 (1996).

131. Robert A. Gorman, *Comments on a Manifesto Concerning the Legal Protection of Computer Programs*, 5 ALB. L.J. SCI. & TECH. 277 (1996).

132. *See* Howard C. Anawalt, *Special Case Note Follow-Up:* Lotus Development Corp. v. Borland International, Inc., 116 S.Ct. 804 (1996), 12 SANTA CLARA COMPUTER & HIGH TECH. L.J. 489, 498 (1996) ("[T]he decision falls far short of being 'definitive' regarding the effect of §102(b) as a limitation on the scope of copyright"); Bradley W. Grout, Note, *Wobbling on the Shoulders of Giants: The Supreme Court's Failure in* Lotus v. Borland, 4 J. INTELL. PROP. L. 77 (1996); Christopher Kanagawa, Note, Lotus v. Borland*: Confusion Within the Computer Industry "Affirmed" By The Supreme Court*, 32 TULSA L. J. 633 (1997); Marci A. Hamilton & Ted Sabety, *Computer Science Concepts in Copyright Case:*

matter of "heated debate,"[133] and that same year the Seventh Circuit, via Judge Easterbrook, issued its decision expressly protecting taxonomies.[134] Meanwhile, additional circuit courts confirmed the principle of *Johnson Controls*, *Whelan*, and *Altai* that the scope of copyright protection extends to nonliteral elements of software.[135]

This brings us to the Java API declarations at issue in *Oracle v. Google* and whether there was something unusual about this decision when assessed pursuant to a case-specific idea/expression dichotomy approach rather than a categorical exclusion for nonliteral elements approach. The developers could have designed the APIs in a countless number of ways but designed an organization they believed later programmers would find both coherent and easy to learn and remember. They made thousands of design choices to create an intuitive and expressive structure that taught the use of the pre-programmed packages of code. The Federal Circuit found that "designing the [APIs] was a creative process," "the declaring code could have been written and organized in any number of ways," and "the Sun/Oracle developers had a vast range of options for the structure and organization."[136] This is far beyond the requirements set out in *Feist*.[137] Indeed, Google hardly disagreed when its Java expert stated that API design "is an art, not a science," distinguished by "the complexity of figuring out how best *to express* what it is that the programmer wants done."[138] There is no doubt that the declaring code is "extremely expressive," and that it reflects creative choices shaped by aesthetic judgments.[139]

That the protection of particular software elements might be subject to a context-sensitive and at least partly factual analysis seems to be a source of great frustration to academics such as Professors Menell and Samuelson, who have consistently urged the adoption of bright line rules against such protection. They argue that industry, and software engineers in particular, must have a bright line rule. They would

---

The Path to a Coherent Law, 10 HARV. J.L. & TECH. 239 (1997); Lloyd L. Weinreb, *Copyright for Functional Expression*, 111 HARV. L. REV. 1149, 1251 (1998) (criticizing copyright law as inconsistent while opining that it has contributed to the extraordinary success of copyright-based industries in the United States).

133. Dennis S. Karjala, *Copyright Protection of Computer Program Structure*, 64 BROOK. L. REV. 519 (1988).

134. American Dental Ass'n v. Delta Dental Plans, 126 F.3d 977 (7th Cir. 1997).

135. General Universal Systems, Inc. v. Lee, 379 F.3d 131 (5th Cir. 2004); BUC Int'l. Corp. v. Int'l Yacht Council Ltd., 489 F.3d 1129 (11th Cir. 2007).

136. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1357, 1359 (Fed. Cir. 2014).

137. Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340 (1991).

138. Trial Tr. at 1004:21–24, 1008:9–14 (May 13, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); Trial Ex. 877, *Oracle v. Google* (No. C 10-03561 WHA).

139. *See supra* note 138; Trial Tr. at 1454:24–1455:3; 1460:1–22 (May 17, 2016), Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

seemingly limit the scope of copyright protection in software to little more than an anti-counterfeiting rule, if that.

In no other industry subject to copyright legal norms has the lack of a bright line rule in separating idea from expression, or fair use from foul, thwarted progress in the manner apparently feared by those advocating for a bright line rule. To the contrary, vast troves of content have been developed even though the legal world attempted to separate idea from expression by a boundary that cannot always be reliably fixed in advance. The number of screenplays, paintings, sculptures, books, short stories, and motion pictures developed since *Nichols v. Universal Pictures Corp*.[140] is astronomical. The amount of software created since 1980 is likewise enormous. [141] Such growth is not because the industry has been operating pursuant to a bright rule on the idea versus expression dichotomy that has now suddenly been disrupted. If anything, the only clear legal consensus is that reflected in the statute: computer programs are protected by copyright. And industry at all levels, from individual app writers to Fortune 100 companies, responded by investing.

Nor did the decision in *Oracle v. Google* seemingly inhibit the development of APIs, as the number of published APIs at one of the web's leading repositories continued to grow at a remarkably rapid pace after May of 2014.

140. Nichols v. Universal Pictures Corp., 45 F.2d 119, 120 (2d Cir. 1930).

141. *See The U.S. Software Industry: An Engine for Economic Growth and Employment*, SOFTWARE AND INFORMATION INDUSTRY ASSOCIATION (2014), https://www.siia.net/Admin/FileManagement.aspx/LinkClick.aspx?fileticket=ffCbUo5PyEM%3D [https://perma.cc/SN2Y-EMQR].

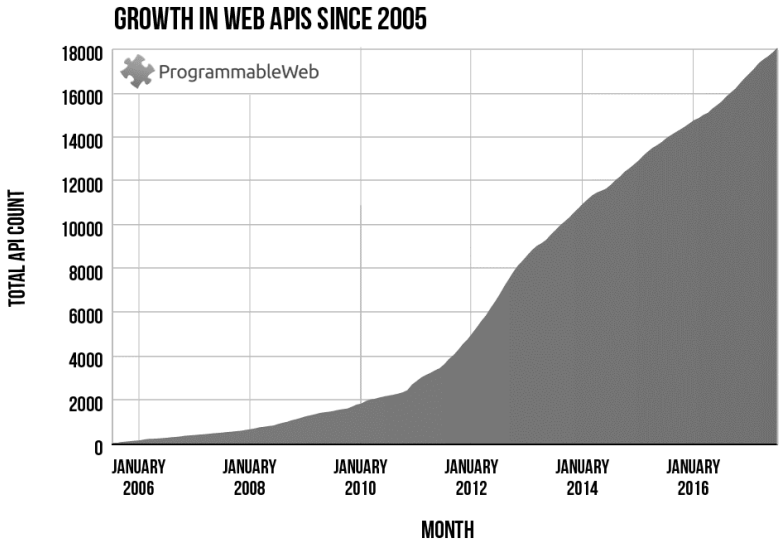

Figure 2: Web APIs published on ProgrammableWeb.[142]

So, the report of API copyright's death is greatly exaggerated, and the report that software generally and APIs specifically will die if they enjoy copyright protection is likewise exaggerated. While it is certainly more interesting to broadly prescribe policy and to do so in adjudicative-sounding legal rules, such is not supported by the sensitive case-specific applications of the idea/expression dichotomy and fair use doctrine. Instead of being a radical departure, *Oracle v. Google* continues the prevailing approach announced more than eighty years ago by Judge Hand in *Nichols*.

142. *ProgrammableWeb Research Center*, PROGRAMMABLEWEB, https://www.programmableweb.com/api-research (last visited Jan. 27, 2018, 3:03PM).