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

RISE OF THE API COPYRIGHT DEAD?: AN UPDATED EPITAPH FOR COPYRIGHT PROTECTION OF NETWORK AND FUNCTIONAL FEATURES OF COMPUTER SOFTWARE

*Peter S. Menell**

TABLE OF CONTENTS

---

* Koret Professor of Law and Director of the Berkeley Center for Law & Technology, University of California at Berkeley School of Law. I thank Jonathan Band, Dylan Hadfield-Menell, Justin Hughes, Mark Lemley, David Nimmer, Tim Simcoe, and Christopher Yoo as well as participants at various presentations (Boston University School of Law, Harvard Law School, New York University School of Law, University of Pennsylvania School of Law, U.C. Berkeley School of Law, U.C.L.A. School of Law) for comments on earlier drafts. Alex Barata, Louise Decoppet, Amit Elazari, and Andrea Hall provided excellent research assistance.

I. INTRODUCTION

As the great Yogi Berra redundantly said, "It's like déjà vu all over again."[1] For IP scholars and practitioners of my generation, Oracle Corporation's lawsuit alleging that Google's Android mobile platform infringes copyright in the Java application program interface ("API") elements has been a stroll down memory lane.[2] Or perhaps less nostalgically for those in the software industry, a zombie horror film set in Silicon Valley.[3]

---

1. *See* YOGI BERRA, THE YOGI BOOK: I DIDN'T SAY EVERYTHING I SAID 9 (1998) (explaining that the déjà vu quotation was inspired by Yankees' sluggers Mickey Mantle and Roger Maris's repeated back-to-back home runs in the early 1960s).

2. As Judge Alsup noted in an early ruling in the *Oracle* litigation, "[t]he term API is slippery." *See* Order Partially Granting and Partially Denying Defendant's Mot. for Summary Judgment on Copyright Claim at 4, Oracle Am., Inc. v. Google Inc., 810 F. Supp. 2d 1002, 1007 (N.D. Cal. 2011) (No. C 10-03561 WHA) (2011 WL 5576228). We will examine the varying and evolving meaning of API throughout this journey.

3. *Cf. List of Zombie Films*, WIKIPEDIA, https://en.wikipedia.org/wiki/List_of_zombie_films [https://perma.cc/TD6M-B36U]. Commentary and news reporting of the *Oracle* case spoke in dire terms. *See, e.g.*, Steven J. Vaughan-Nichols, *Oracle v. Google, and the End of Programming as We Know It*, COMPUTERWORLD (May 16, 2016), http://www.computerworld.com/article/3070001/application-development/oracle-v-google-and-the-end-of-programming-as-we-know-it.html [https://perma.cc/SY5L-WPZC]; Klint Finley, *The Oracle-Google Case Will Decide the Future of Software*, WIRED (May 23, 2016), http://www.wired.com/2016/05/oracle-google-case-will-decide-future-software/ [https://perma.cc/6U69-YGJW] (opining that "nothing less is at stake [in the outcome of the

I cut my teeth analyzing the scope of copyright protection for network and other functional features of computer software. My first foray into intellectual property scholarship examined the interplay among the utilitarian nature of computer programming, the distinctive network economics of software markets, and the role of copyright protection within the larger intellectual property system.[4] Along with other scholars and practitioners,[5] I wrote about and filed amicus briefs in battles over interoperability,[6] reverse engineering,[7] graphical user interfaces,[8] and menu command hierarchies.[9] After more than a dec-

Oracle v. Google litigation] than the future of programming"); Joe Mullin, *Second* Oracle v. Google *Trial Could Lead to Huge Headaches for Developers*, ARS TECHNICA (May 8, 2016), http://arstechnica.com/tech-policy/2016/05/round-2-of-oracle-v-google-is-an-unpredictable-trial-over-api-fair-use/ [https://perma.cc/F8FQ-SAY9] (reporting that if those who develop APIs "can use copyright law to control how programming is done, there will be a sea change in industry practices. For many developers, especially of open source software, this will be a change for the worse.").

4. *See generally* Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329 (1987) (analyzing legal protection for computer software based on my third-year paper at Harvard Law School); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989); Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644 (1994); Dennis S. Karjala & Peter S. Menell, *Applying Fundamental Copyright Principles to* Lotus Dev. Corp. v. Borland Int'l, Inc., 10 HIGH TECH. L.J. 177 (1995).

5. Professors Dennis Karjala, Jerome Reichman, and Pamela Samuelson, copyright treatise authors Paul Goldstein and David Nimmer, practitioners Jonathan Band, Peter Choy, David Hayes, Michael Jacobs, Gary Reback, and Richard Stern, economists Joseph Farrell and Brian Kahin, and computer scientist Randal Davis were among the early fellow travelers. The network economics research of Professors Joseph Farrell, Michael Katz, Garth Saloner, and Carl Shapiro provided valuable insights.

As the first wave of copyright API litigation was building, Professor Karjala, Professor Samuelson, and I convened a broad range of intellectual property scholars, practitioners, software experts, and economists to examine the emerging issues and jurisprudential puzzles. That conference produced a consensus report among the legal academics that helped clarify key software copyright issues and foreshadowed important legal developments. *See generally* Donald S. Chisum, Rochelle Cooper Dreyfuss, Paul Goldstein, Robert A. Gorman, Dennis S. Karjala, Edmund W. Kitch, Peter S. Menell, Leo J. Raskind, Jerome H. Reichman & Pamela Samuelson, *LaST Frontier Conference on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15 (1989) [hereinafter *LaST Frontier Software Report*]. In addition, I advised the U.S. Congress's Office of Technology Assessment, which produced several useful reports. *See* OFFICE OF TECH. ASSESSMENT, OTA-TCT-527, FINDING A BALANCE: COMPUTER SOFTWARE, INTELLECTUAL PROPERTY, AND THE CHALLENGE OF TECHNOLOGICAL CHANGE (1992), http://ota.fas.org/reports/9215.pdf [https://perma.cc/FGT7-973D]; OFFICE OF TECH. ASSESSMENT, U.S. CONG., OTA-BP-CIT-61, COMPUTER SOFTWARE AND INTELLECTUAL PROPERTY: BACKGROUND PAPER (1990), http://ota.fas.org/reports/9009.pdf [https://perma.cc/E4DC-9YMB].

6. *See* Comput. Associates Int'l v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240 (3d Cir. 1983).

7. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); *see also* Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.2d 596 (2000); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1540 (11th Cir. 1996) (following Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992)).

8. *See* Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.2d 1435 (9th Cir. 1994); *see also* Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204 (9th Cir. 1988).

ade of software copyright wars,[10] the hostilities ceased following the resolution of the epic battle between Lotus and Borland over the spreadsheet menu command hierarchy.[11] To mark closure of that era, I wrote an "epitaph" for copyright protection of network features of computer software.[12]

Although the Supreme Court deadlocked over the *Lotus v. Borland* appeal,[13] the computer industry achieved *détente* following several lower-court cases rejecting copyright protection for APIs and other high-level, functional features of computer software. Congress reinforced these principles in crafting the anti-circumvention provisions of the Digital Millennium Copyright Act of 1998 ("DMCA").[14] This is not to say that copyright law does not protect computer software, but rather that the scope of protection is narrow and focused on purely expressive or arbitrary — as opposed to functional — elements of computer programs.

Veterans of the API copyright battles moved on to new software IP battlefronts. Microsoft's anti-competitive practices in the "browser wars" emerged as a new battleground in the late 1990s.[15] One flank

---

9. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202 (D. Mass 1993), 831 F. Supp. 223 (D. Mass. 1993), *rev'd* 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

10. *See* JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL: INTELLECTUAL PROPERTY AND INTEROPERABILITY IN THE GLOBAL SOFTWARE INDUSTRY (1995); Nell Margolis, *Users Biggest Losers in Spreadsheet Wars*, 29 COMPUTERWORLD 8 (July 16, 1990) (commenting on the district court ruling finding copyright infringement in *Lotus v. Borland*). Sixteen years later, Band and Katoh published a retrospective exploring the enactment of the DMCA and implementation of its interoperability provisions and international developments. It also touches on patent and antitrust issues. *See* JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL 2.0 (2011). Band and Katoh wrote the book before the *Oracle v. Google* case triggered the second wave of copyright API litigation.

11. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

12. *See* Peter S. Menell, *An Epitaph for Traditional Copyright Protection of Network Features of Computer Software*, 43 ANTITRUST BULL. 651 (1998).

13. Justice Stevens recused himself. *See* David Einstein, *Borland Bests Lotus in 6-Year Legal Battle*, S.F. GATE (Jan. 17, 1996), http://www.sfgate.com/business/article/Borland-Bests-Lotus-in-6-Year-Legal-Battle-2998221.php [https://perma.cc/BB5C-SL4S] (reporting that Justice Stevens recused himself because of his ownership of IBM stock). In view of his intellectual property jurisprudence, as reflected in his opinions Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417 (1984) and Parker v. Flook, 437 U.S. 584 (1978), he likely would have joined the four justices voting to affirm the First Circuit's decision.

14. *See* 17 U.S.C. § 1201(f) (2012) (interoperability exception for anti-circumvention provisions); *see also id.* at § 1201(a) (exemption process). For an explanation of the Congressional intent behind these provisions, *see infra* notes 165–68.

15. I consulted for a consortium of State Attorneys General for nearly a decade on that battle and its aftermath. *See* STATE OF CA, DEP'T OF JUST., OFFICE OF THE ATT'Y GEN., *Antitrust Highlights*, https://oag.ca.gov/antitrust/highlights [https://perma.cc/6KBV-L997]; Stephen D. Houck & Kevin J. O'Connor, *Comments on the States' Role in the Microsoft Case Re: Working Group on Enforcement Institutions* (2007); New York v. Microsoft Corp., 224 F. Supp. 2d 76 (D.D.C. 2002); *see generally United States v. Microsoft Corp.*,

touched on API copyright protection. Sun Microsystems sued Microsoft over breach of contract and copyright infringement relating to Microsoft's forking[16] of Sun's Java™ software platform. That litigation settled with Microsoft paying Sun $20 million, and Sun chose not to assert its copyright infringement claims in court.[17] The conduct at issue also contributed to Sun's later antitrust and patent infringement lawsuit against Microsoft, which resulted in a $1.6 billion settlement.[18]

By the late 1990s, the open source movement was gaining momentum, further reducing the use of proprietary strategies in the development of APIs. Sun released the core Java language for use by programmers, although it sought to ensure that the Java platform remained interoperable across different systems. Following the burst of the dot-com bubble in the 2000–2002 period, software patent assertion added a new dimension to software litigation. Standard setting organizations ("SSOs") emerged as a principal bulwark in promoting interoperable interface development.[19]

By the early 2000s, software copyright disputes, and particularly those relating to APIs, were rare. Although interoperability skirmishes occasionally flared,[20] the copyright jurisprudence remained remarkably stable. Silicon Valley moved on, or so many of the API copyright veterans thought. Much of the API action shifted to the patent and

---

WIKIPEDIA, https://en.wikipedia.org/wiki/United_States_v._Microsoft_Corp. [https://perma.cc/H7HS-K9B3].

16. Forking of software code refers to creating an independent branch of a computer program. *See Fork (Software Development)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Fork_(software_development) [https://perma.cc/KN73-ALQQ]. This split from the original program typically "spawns competing projects that cannot later exchange code, splitting the potential developer community." Eric S. Raymond, *Promiscuous Theory, Puritan Practice*, *in* HOMESTEADING THE NOOSPHERE (2002), http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/ar01s03.html [https://perma.cc/RN3F-7R89].

17. I advised Sun Microsystems' legal team about copyright's limiting doctrines in 1999. I was relieved to see the API copyright claims die a quiet death. *See infra* notes 245–56.

18. *See* Scarlet Pruitt & Paul Roberts, *Microsoft to Pay $700 Million for Antitrust Issues, $900 Million to Resolve Patent Dispute*, INFOWORLD (Apr. 2, 2004), http://www.infoworld.com/article/2667124/operating-systems/update--sun--microsoft-settle-suit-in-billion-dollar-pact.html [https://perma.cc/2Y6D-ZAS5].

19. *See* Jorge L. Contreras, *Patents, Technical Standards and Standard-Setting Organizations: A Survey of the Empirical, Legal and Economics Literature*, *in* RESEARCH HANDBOOK ON THE ECONOMICS OF INTELLECTUAL PROPERTY LAW VOL. 2 — ANALYTICAL METHODS (Peter S. Menell & David Schwartz eds., forthcoming 2018); Mark A. Lemley, *Intellectual Property Rights and Standard-Setting Organizations*, 90 CALIF. L. REV. 1889 (2002).

20. *See, e.g.*, Patrick Mannion, *Ruling for Green Hills Clears Way for Copying of APIs*, EE TIMES (Aug. 21, 2007), http://www.eetimes.com/document.asp?doc_id=1166905 [https://perma.cc/ZW7L-TGFH] (reporting that the arbitration panel held that copyright laws do not extend to the functionality of APIs in a dispute involving real time operating systems). I served as an expert witness for Green Hills in the case.

standard setting realms.[21] Internet piracy emerged as the major copyright battleground, and a new war — between Hollywood and Silicon Valley — took center stage.[22]

Then a startling new API copyright case made headlines in August 2010.[23] In January of that year, Oracle Corporation acquired Sun Microsystems for $5.6 billion.[24] In August, Oracle sued Google for patent and copyright infringement over the Android platform, one of the two leading mobile computing platforms (Apple's iOS was the other).[25] Google built Android using the Java programming language and declarations — headers that name and describe functions — from 37 of the 166 "packages" of the Java™ Platform, Standard Edition API Specification.[26] Oracle would ultimately seek over $9 billion in damages and an injunction blocking Google's use of Android.[27]

The API copyright resurgence is not limited to *Oracle v. Google*. In 2014, Cisco Systems, a leading manufacturer of networking equipment, sued Arista Networks for patent and copyright infringement.[28] The copyright claims focused on Cisco's command line interface ("CLI") for configuring, monitoring, and maintaining Cisco

---

21. *See* Jorge L. Contreras, *A Brief History of FRAND*, 80 ANTITRUST L.J. 39 (2015); Peter S. Menell & Michael J. Meurer, *Notice Failure and Notice Externalities*, 5 J. LEGAL ANALYSIS 1 (2013); Peter S. Menell, *Forty Years of Wondering in the Wilderness and no Closer to the Promised Land:* Bilski*'s Superficial Textualism and the Missed Opportunity to Return Patent Law to its Technology Mooring*, 63 STAN. L. REV. 1289 (2011).

22. *See* Peter S. Menell, *Envisioning Copyright Law's Digital Future*, 46 N.Y.L. SCH. L. REV. 63 (2002).

23. *See* Don Clark & Cari Tuna, *Oracle Suit Challenges Google — Silicon Valley Giants Tangle Over Patents, Copyrights Involving Open Programs Android and Java*, WALL ST. J. B1 (Aug. 13, 2010) (noting that the lawsuit was a "surprise move" and "set off shock waves in the Silicon Valley software community"); *see also* Cari Tuna & Don Clark, *Oracle's Java Suit Gives a Jolt*, WALL. ST. J. B1 (Aug. 14, 2010) (reporting that "[l]awyers and software developers were scrambling Friday to analyze whether other Java-based products might run afoul of Oracle's intellectual property — and if legal risks may extend to a broader array of what the industry calls open-source software").

24. *See Sun Acquisition by Oracle*, WIKIPEDIA, https://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle [https://perma.cc/B57Z-WLZW]. The parties agreed to the acquisition in April 2009. *Id.* Due to regulatory approvals, the transfer did not occur until January 2010. *Id.* The sale price was $7.4 billion, resulting in a net price of $5.6 billion after accounting for Sun's cash and debt. *Id.*

25. *See* Eric Bangeman, *Oracle Sues Google Over Use of Java in Android*, ARS TECHNICA (Aug. 12, 2010), https://arstechnica.com/tech-policy/2010/08/oracle-sues-google-over-use-of-java-in-android-sdk/ [https://perma.cc/W4LN-D34L].

26. These packages are compilations of functions. *See infra* notes 239–40, 249, 322 and accompanying text.

27. *See* Joe Mullin, *Oracle Will Seek a Staggering $9.3 Billion in 2nd Trial Against Google*, ARS TECHNICA (Mar. 29, 2016), http://arstechnica.com/tech-policy/2016/03/oracle-will-seek-a-staggering-9-3-billion-in-2nd-trial-against-google/ [https://perma.cc/ZB8E-WK7Y]; Daniel Siegal, *Oracle, Google File Heated Trial Briefs In $8B IP Showdown*, LAW360 (Apr. 21, 2016), https://www.law360.com/articles/787442/oracle-google-file-heated-trial-briefs-in-8b-ip-showdown [https://perma.cc/BX2C-VMVF].

28. *See* Quentin Hardy, *In Suit, Cisco Accuses Arista of Copying Work*, N.Y. TIMES (Dec. 5, 2014), http://bits.blogs.nytimes.com/2014/12/05/in-suit-cisco-accuses-arista-of-copying-work/ [https://perma.cc/K79H-C2JK].

devices.[29] Arista, formed by a Cisco founder and employing many former Cisco engineers, designs and sells competing network switches. Arista allegedly copied more than five hundred of Cisco's CLI commands in developing its EOS network operating system.[30]

With these headlines, I was beginning to feel a bit like the aging Michael Corleone, as portrayed by Al Pacino, in *The Godfather: Part III*: "Just when I thought I was out . . . they pull me back in."[31] As this Article explains, the new wave of API litigation is not entirely "déjà vu all over again." *Oracle v. Google* involves a more complex interface specification than those involved in the first wave of cases. And unlike defendants in those cases, Google did not seek to achieve complete end-user interoperability. Rather, Google developed a new operating system that selected from and augmented the Java API packages to optimize a powerful new mobile platform for smartphones. Google also used a more permissive licensing model than Sun and Oracle used for the Java platform.

Although achieving complete end-user interoperability is a functional objective that can serve to limit copyright protection, it is not the sole limiting rationale for excluding functional features and function labels from copyright protection. The principles explicated in my first Epitaph apply with equal force to this newer API copyright wave. Fundamental copyright doctrines circumscribe protection for APIs.

This Article updates and expands upon the earlier Epitaph to address the second API copyright wave. As background, Part II reviews the first wave of API copyright legislation and litigation. Part III examines the *Oracle v. Google* litigation. Part IV critically analyzes the *Oracle v. Google* litigation and explains that copyright law's fundamental exclusion of protection for functional features dictates that the labeling conventions and packaging of functions within interface specifications generally fall outside of the scope of copyright protection even though the implementing code garners protection. This interpretation of copyright law serves the larger goals of intellectual property law and competition policy.

The technological, legal, and factual complexity of this drama requires familiarity with various technical terms and storylines. Appen-

---

29. *Id.*

30. *See* Second Amended Complaint for Copyright and Patent Infringement, Cisco Sys. Inc v. Arista Networks, Inc., No. 14-cv-05344-BLF, 2016 WL 632000 (N.D. Cal. Feb. 17, 2016); Jeffrey Burt, *Cisco Sues Networking Rival Arista in Patent Dispute*, EWEEK (Dec. 5, 2014), http://www.eweek.com/networking/cisco-sues-neworking-rival-arista-in-patent-dispute.html [https://perma.cc/P68R-PYJ3] (quoting Mark Chandler, Cisco's Senior Vice President and General Counsel, pointing to the copying of more than 500 multi-word command-line expressions in Arista's EOS operating system).

31. *See The Godfather: Part III (1990) — Quotes*, IMDB, http://www.imdb.com/title/tt0099674/quotes [https://perma.cc/4NN8-4W7N]; *Just when I thought I was out . . . they pull me back in*, YOUTUBE, https://www.youtube.com/watch?v=UPw-3e_pzqU (last visited Jan. 27, 2018).

dix A provides a glossary of key terms. Appendix B identifies the key corporate and individual participants. Appendix C provides a comprehensive timeline. Appendix D summarizes the 37 APIs at issue. Appendix E traces the fair use trial.

II. COPYRIGHT PROTECTION FOR COMPUTER SOFTWARE 1.0

The first wave of computer software litigation frames the modern API battlefront. Section A begins with a personal account, which highlights the emergence of the API copyright issue and puts the first wave of API copyright jurisprudence in proper perspective. Section B sets the stage for the decade-long API copyright wars, surveying the copyright law background, the economics of interoperability, and the industrial backdrop. Section C traces the API copyright protection battlefront in the courts, Congress, and the Copyright Office. It examines the major software cases. The final Section summarizes the resolution of the API copyright wars and how this era reinforced the underlying logic of the intellectual property system.

A. A Personal Account

I encountered the economic effects of legal protection for computer software in a serendipitous way while pursuing graduate degrees in economics and law in the early 1980s. While completing my Ph.D. dissertation, I faced a familiar formatting challenge: incorporating integral signs and other mathematical symbols into dissertation chapters. Mainframe computer technology offered symbolic notation tools, but that required periodic trips to Stanford's Forsythe Hall to retrieve printouts on the central laser printer. Traveling across campus only to find a large printout with the words "SYNTAX ERROR" was frustrating. There had to be a better way.

I was excited to learn that XyQuest had introduced a computer program, XyWrite, which coded symbolic notation for the newly introduced IBM desktop personal computer ("PC"). It offered the capability of printing drafts at the touch of a button on a convenient dot matrix printer attached to the desktop computer. Unfortunately, the cost of the system was well beyond my means. IBM was charging three thousand dollars for the PC.

As a microcomputer hobbyist, I was aware that IBM did not manufacture many of the PC's components. Tandem, for instance, made the disk drives, while Amdec made the monitor. Advertisements in the back of computer magazines revealed that I could assemble much of the IBM PC for a fraction of its retail price. After IBM began selling the stripped-down PC chassis and main boards to university students at a steep discount, I assembled a fully functional IBM PC at

about half the retail price. To a graduate student studying microeconomic theory, industrial organization, and antitrust policy, this price differential posed a puzzle.

Reverting to my rudimentary legal training, I traced the source of IBM's extraordinary market power to trade secret and copyright protection over the Basic Input/Output System ("BIOS") firmware interface — not a particularly innovative piece of the overall computer architecture, but a critical component for interoperability. Combining law and economics, I came to see that expansive copyright protection for computer software could undermine both rapid innovation and realization of positive network effects, and conflicted with the logic of the intellectual property system.[32]

Copyright's foundational idea-expression doctrine and independent creation defense provided key pieces to solving the puzzle and ultimately proved IBM's undoing.[33] Within a few years, Phoenix and Compaq reverse engineered the IBM PC BIOS and developed much less expensive interoperable "clones" that displaced IBM's dominance.[34] Microsoft, which controlled the leading microcomputer operating systems (DOS and later Windows) and mastered the economics of interoperability, would become the dominant computer company over the next two decades.

B. Setting the Stage

In order to appreciate the API copyright controversy, it is important to understand the intellectual property landscape that existed when the software marketplace took flight in the early 1980s, the economics of interoperability, and the software industry.

---

32. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4.

33. After the emergence of home computers designed and built by start-ups for computing hobbyists in the late 1970s, IBM skyrocketed to dominance with the launch of its PC line of microcomputers for home and business use. *See* Andrew Pollack, *Big I.B.M. Has Done It Again*, N.Y. TIMES, Mar. 27, 1983, http://www.nytimes.com/1983/03/27/business/big-ibm-has-done-it-again.html (last visited Jan. 27, 2018) (reporting that by 1983, "[v]irtually every software company [was] giving first priority to writing programs for the I.B.M. machine"); *Personal Computers: and the Winner is IBM*, BUS. WK., Oct. 3, 1983, at 76; *IBM's Personal Computer Spawns an Industry*, BUS. WK., Aug. 15, 1983, at 88.

34. *See* Sam Whitmore, *PC-Compatible ROM BIOS Emerges from Phoenix*, PC WK., May 8, 1984, at 5; Leslie Helm, *IBM's 'Clone Killers' Don't Scare Phoenix Technologies*, BUS. WK., Dec. 21, 1987, at 113; *see generally* Steven Burke, *Court Support for 'Clean Room' Cloning May Legalize Intel '386 Chip' Work-Alikes*, PC WK., Feb. 27, 1989, at 63; Russell Moy, *A Case Against Software Patents*, 17 SANTA CLARA COMPUTER & HIGH TECH. L.J. 67, 70–73 (2000) (chronicling reverse engineering of the IBM BIOS).

1. The Intellectual Property Backdrop: Legislation and Legislative History

Computer software, by its very nature as written work intended to serve utilitarian purposes, defies easy categorization within the intellectual property system.

As the computer software marketplace emerged in the early 1970s, policymakers faced a dilemma. Computer software could be expensive to develop and was easily pirated, creating a severe appropriability problem for the nascent, yet critical, software industry.[35] Patent law, which had long served as the primary form of protection for technological advances in machines and processes, was thought to be too costly, time-consuming, stringent, and uncertain a means for protecting software products against piracy.[36] Copyright law had long provided an effective means of protecting literary works from piracy, but its doctrines excluding ideas and functional elements from protection[37] raised serious questions about its appropriateness for protecting inherently utilitarian works. Copyright's low threshold for protection,[38] complex scope,[39] broad array of rights,[40] and long duration[41] created a risk of overbroad protection for computer software products.

The software protection controversy also emerged at an inopportune time. Congress had been working for nearly two decades to

35. *See* Bill Gates, *An Open Letter to Hobbyists*, LETTERS OF NOTE (Feb. 3, 1976), http://www.lettersofnote.com/2009/10/most-of-you-steal-your-software.html [https://perma.cc/H7E6-H8NK] (an angry letter written by a young Bill Gates complaining about widespread piracy of Microsoft's first software product — Altair BASIC, written by Bill Gates, Paul Allen, and Monte Davidoff: "As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if people who worked on it get paid? Is this fair?").

36. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4, at 1347–51.

37. *See* Baker v. Selden, 101 U.S. 99, 102 (1879) (differentiating the scope of copyright and patent:

> To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.

).

38. *See* Feist Publ'ns, Inc. v. Rural Tel. Service Co., Inc., 499 U.S. 340 (1991).

39. *See* PETER S. MENELL, MARK A. LEMLEY & ROBERT P. MERGES, INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE: 2017, VOL II: COPYRIGHTS, TRADEMARKS & STATE IP PROTECTIONS, ch. IV(E) (2017).

40. *See id.*; 17 U.S.C. § 106(2) (2012) (codifying the right to prepare derivative works).

41. At the time, copyright protection lasted for 56 years from publication, whereas patent protection lasted for 17 years from grant. Congress planned to expand copyright duration significantly (to life of the author plus 50 years or 75 years in the case of entity authors) at the time that the software protection issue arose.

overhaul the Copyright Act of 1909 and was nearing closure in the mid-1970s.[42] Faced with the challenge of fitting computer software and other new information technologies under the existing umbrella of intellectual property protection, Congress established the National Commission on New Technological Uses of Copyrighted Works ("CONTU") to study the implications of the new technologies and recommend revisions to federal intellectual property law.[43] As a stopgap, Congress included computer software within the scope of "literary works" in the Copyright Act of 1976 ("1976 Act").[44] Other provisions of the 1976 Act, however, maintained traditional exclusions for ideas and functional features.[45]

After conducting extensive hearings and receiving expert reports, a majority of CONTU's blue-ribbon panel of copyright authorities and interest group representatives concluded that the intellectual work embodied in computer software should be protected under copyright law, notwithstanding the fundamental principle that copyright cannot protect "any idea, procedure, process, system, method of operation, concept, principle, or discovery"[46] and the Supreme Court's foundational decision on the idea-expression dichotomy in *Baker v. Selden*.[47]

---

42. *See* Peter S. Menell, *In Search of Copyright's Lost Ark: Interpreting the Right to Distribute in the Internet Age*, 59 J. COPYRIGHT SOC'Y U.S.A. 1 (2011).

43. Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873 (1974).

44. The Act includes "literary works" within the class of "works of authorship." *See* 17 U.S.C. § 102(a)(1) (2012). The House Report explains that "[t]he 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*." H.R. REP. NO. 94-1476, at 53–54 (1976) (emphasis added).

45. *See* 17 U.S.C. § 102(b) (2012) ("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."); *id.* at § 101 (

> Pictorial, graphic, and sculptural works' include two-dimensional and three-dimensional works . . . Such works shall include works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned; the design of a useful article, as defined in this section, shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article

); *id*. ("A 'useful article' is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. An article that is normally a part of a useful article is considered a 'useful article.'").

46. 17 U.S.C. § 102(b) (2012).

47. Baker v. Selden, 101 U.S. 99 (1879). *See* NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 1 (1979) [hereinafter CONTU REPORT]; *but see id.* at 27–37 (Commissioner Hersey, dissenting) (arguing that "forcible wrenching" would be required to protect computer programs under the copyright law); *id.* at 37–38 (Commissioner Karpatkin, dissenting) (same); *cf. id.* at 26–27 (Commissioner Melville Nimmer, concur-

CONTU recommended two modest changes to the 1976 Act: (1) adding a definition for computer programs — "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"; and (2) expressly immunizing "the rightful possessor of a copy of a computer program" from infringement liability for running and making a backup copy of the program.[48] Congress implemented CONTU's recommendation in its 1980 amendments to federal copyright law with a confusing wording change.[49]

The CONTU Final Report explained that while "one is always free to make a machine perform any conceivable process (in the absence of a patent), . . . one is not free to take another's program," subject to copyright's limiting doctrines, originality and the idea-expression dichotomy.[50] The Report further explained that:

> The 'idea-expression identity' exception provides that copyrighted language may be copied without infringing when there is but a limited number of ways to express a given idea. This rule is the logical extension of the fundamental principle that copyright cannot protect ideas. In the computer context this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement.[51]

Thus, while recognizing important limitations on copyright protection for computer software, including the § 102(b) limitations, Congress intended that software programmers would garner protection for their program design and coding choices to the extent that the expression was separable from the underlying ideas. In this way, the general programming ideas and unoriginal programming choices remain free for others to use while the creative effort in particularized

---

ring) (warning that CONTU recommendations might take copyright law "beyond the breaking point," converting it into a general misappropriation law).

48. *See* CONTU REPORT at 12.

49. Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3007, 3028 (1980) (codified at 17 U.S.C. §§ 101, 117 (2012)). For reasons that were not explained in the legislative history of the 1980 amendments, Congress narrowed CONTU's category of "rightful possessor" to "rightful owner." *See* 2 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 8.08(B)(1)(c)(ii) (2017).

50. *See* CONTU REPORT at 20. Courts have treated the CONTU REPORT as legislative history to the 1980 amendments to the 1976 Act. *See* Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 260–61 (5th Cir. 1988); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1252 (3d Cir. 1983).

51. CONTU REPORT at 20 (footnote omitted).

programming choices and compilations, especially in complex programs, gains protection against copyists.

2. Network Economics

The computer and software industries ushered in a revolutionary economic era. Whereas major conventional markets — from automobiles to conventional appliances, raw materials, food, and consumer products — have thrived on competition among many suppliers, computer hardware and software markets tended toward one or a few dominant players for a distinctive reason: consumers, programmers, and system users care about network effects. They want to communicate among devices and among software products running on their devices. They care about interoperability — among hardware devices, between software and hardware devices, and across software. They value the investment that they have made in learning software interfaces. Once consumer or programmer bandwagons take hold, markets tip decisively toward an emerging dominant platform.

Robert Metcalfe, a co-inventor of Ethernet,[52] captured this dynamic in simple mathematical and economic terms: "the value of a telecommunications network is proportional to the square of the number of connected users of the system."[53] Like human languages, common (and interoperable) computer languages and interfaces are incredibly important. Such network effects have come to dominate computer hardware, software, and Internet markets.

Network effects generated new strategies among computer hardware and software companies. The ability to control interfaces through intellectual property protection, technological protections (such as digital rights management), and contracts became a major part of these industries. Having innovative, competitively-priced products continued to be important, but establishing and building a successful software-based platform became the key to success.[54] Companies could use API strategies to lock in consumers and lock out competitors.

As my anecdote about the IBM PC illustrates,[55] hardware companies with a large installed base of users could attract software developers to write for their platform, thereby generating a virtuous feedback loop — what economists call increasing returns. As more

52. *See Ethernet*, WIKIPEDIA, https://en.wikipedia.org/wiki/Ethernet [https://perma.cc/W94V-58YX].

53. *See Metcalfe's law*, WIKIPEDIA, https://en.wikipedia.org/wiki/Metcalfe%27s_law [https://perma.cc/EMQ4-8DJU].

54. *See* CARL SHAPIRO & HAL R. VARIAN, INFORMATION RULES: A STRATEGIC GUIDE TO THE NETWORK ECONOMY 103–226 (1999) [hereinafter INFORMATION RULES].

55. *See supra* text accompanying notes 32–33.

software became available for the IBM PC, the functionality of the base computer expanded, which spurred greater demand for the IBM PC. This growth motivated programmers to write even more programs for that platform. It was only after Phoenix and Compaq successfully reverse-engineered and produced clean room[56] versions of the IBM BIOS that IBM's hold on the microcomputer marketplace loosened, resulting in robust competition and a dramatic drop in microcomputer prices. Other computer companies used API strategies to control access to their video game platforms, cell phone networks, replacement parts (such as ink cartridges for printers), and graphical user interfaces.[57]

The contours of intellectual property rules governing interoperability strategies — copyright, patent, trade secret, and anti-circumvention laws, as well as the preemption of contractual restrictions — became a major battleground.

3. The Industrial Backdrop

Companies and programmers divided on the proper role of intellectual property protection in controlling APIs. Many established hardware and software entities, such as IBM, Digital Equipment Corporation, Apple Computer Corporation, and Lotus Development Corporation, in conjunction with leading industry trade organizations, such as the Computer and Business Equipment Manufacturers Association ("CBEMA") and the Software Publishers Association, advocated strong copyright protection for computer interfaces.[58]

On the other side, the free and open source software movement, formed through grassroots organizing among programmers and academic researchers who valued collaborative research and sharing of software, opposed intellectual property protection for computer software.[59] These researchers believed proprietary limitations on access to and use of software would undermine freedom and innovation.

Open source software originated in the early 1970s in the culture of collaborative research on computer software that existed in many software research environments.[60] To perpetuate that model in the

56. A clean room process insulates programmers from copyright protected code in producing code that accomplishes the same functions as a target program based solely on the functional specifications. Such a process ensures a program is independently written and hence not copied except with regard to unprotectable elements. *See generally* P. Anthony Sammi, Christopher A. Lisy, & Andrew Gish, *Good Clean Fun: Using Clean Room Procedures in Intellectual Property Litigation*, 25 INTELL. PROP. & TECH. L.J. 3 (2013); *supra* text accompanying note 34.

57. *See generally* INFORMATION RULES, *supra* note 54.

58. *See generally* BAND & KATOH, *supra* note 10, at xvii, 120–22.

59. *See* STEVEN WEBER, THE SUCCESS OF OPEN SOURCE (2004).

60. *See id.*; ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY (1999).

face of increasingly proprietary software, Richard Stallman, a former researcher in MIT's Artificial Intelligence Laboratory, established the Free Software Foundation ("FSF") to promote users' rights to use, study, copy, modify, and redistribute computer programs.[61] Such rights diverge from copyright law's traditional bundle of exclusive rights. For that reason, FSF developed the GNU ("GNU's Not Unix!") General Public License ("GPL"), an unconventional licensing agreement. Also referred to as "copyleft," it is designed to prevent programmers from building proprietary limitations into "free" software.[62] The GPL guarantees end users the freedoms to run, study, share (copy), and modify the software as long as the users permit the use of any derivative works on the same terms.[63] In this way, GPL software "infects" derivative works and spreads, like a virus, through the ecosystem — liberating computer software from proprietary rights.

Stallman set forth a task list for the development of a viable UNIX-compatible open source operating system.[64] Many programmers throughout the world contributed to this effort on a voluntary basis, and by the late 1980s, they had assembled most of the components. The project gained substantial momentum in 1991 when Linus Torvalds developed a UNIX-compatible kernel[65] dubbed "Linux." Torvalds structured the evolution of his component on the GNU GPL open source model. The integration of the GNU and Linux components resulted in a UNIX-compatible open source program, referred to as GNU/Linux, that has since become widely used throughout the computing world.[66] In the process, it spawned a large community of computer programmers and service organizations committed to open source development. The growth and success of Linux brought the open source movement into the mainstream computer software industry.

---

61. *See Richard Stallman*, WIKIPEDIA, https://en.wikipedia.org/wiki/Richard_Stallman [https://perma.cc/CS7R-VKWS].

62. *See GNU General Public License*, WIKIPEDIA, https://en.wikipedia.org/wiki/GNU_General_Public_License [https://perma.cc/P6YD-ZDWR].

63. *See* Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443 (2005).

64. *See GNU Project*, WIKIPEDIA, https://en.wikipedia.org/wiki/GNU_Project. [https://perma.cc/79XW-LVJF]. The UNIX operating system, initially developed by researchers at MIT, AT&T, and General Electric in the late 1960s and early 1970s, became a foundation for modern computer operating system design. *See History of Unix*, WIKIPEDIA, https://en.wikipedia.org/wiki/History_of_Unix [https://perma.cc/9FYB-GCD2]; Marshall Kirk McKusick, *Twenty Years of Berkeley Unix: From AT&T Owned to Freely Redistributable*, *in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION 31, 36–39 (Chris DiBona et al. eds., 1999).

65. The kernel is a computer program that constitutes the central core of a computer's operating system. *See Kernel (operating system)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Kernel_(operating_system) [https://perma.cc/KEF6-NCFB].

66. For example, the Linux kernel is an integral component of the Android operating system. *See Android (operating system)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Android_(operating_system) [https://perma.cc/E6P6-DD2S].

The Computer Systems Research Group ("CSRG") of the University of California, Berkeley spearheaded a parallel effort and ultimately produced another UNIX operating system derivative. Bill Joy, one of the founders of Sun Microsystems, played a key role in the development of Berkeley UNIX.[67] In contrast to the GPL, the Berkeley Software Distribution ("BSD") project offered its software on a less restrictive basis.[68] The distinction between GPL and more permissive open software licenses plays a central role in the second wave of API copyright litigation.[69]

Recognizing the importance of interoperability to consumers, competition, and cumulative innovation, a new generation of technology companies formed the American Committee for Interoperable Systems ("ACIS") in the early 1990s to advocate for less protectionist intellectual property policies for computer software.[70] Sun Microsystems and Oracle were among ACIS's founding members.[71] Peter M.C. Choy, Sun's Deputy General Counsel, served as ACIS's Chairman. In a letter to President-Elect William Clinton's transition team, Choy advocated a scope of copyright protection for computer software "which balances incentives for developers with the public interest in competitiveness, open systems, and incremental innovation. Sun believes, as its fellow members of ACIS believe, the over-protection of technology under intellectual property law may lead . . . to 'monopolistic stagnation' in the industry."[72] Sun and Oracle play a central role in the second wave of API litigation. As explored in Part III, Oracle took a far more protective approach to copyright protection of APIs after its acquisition of Sun.[73]

C. The API Copyright War

These conditions produced a multi-front war over copyright protection for computer software containing features that generate or rely

---

67. *See Berkeley Software Distribution*, https://en.wikipedia.org/wiki/Berkeley_Software_Distribution; [https://perma.cc/LD5K-EFYK]; *Bill Joy*, WIKIPEDIA, https://en.wikipedia.org/wiki/Bill_Joy. [https://perma.cc/3UN3-S6FW].

68. *See Permissive Software Licence*, WIKIPEDIA, https://en.wikipedia.org/wiki/Permissive_software_licence [https://perma.cc/UZ3K-WYY5].

69. *See infra* notes 283, 291, 317 and accompanying text.

70. *See id.*; ACIS, Statement of Principles contained in Attachment to Letter from Peter M.C. Choy to Professor Barry E. Carter (Nov. 5, 1992), https://www.ccianet.org/wp-content/uploads/2014/10/ACIS-Letter-to-Clinton-Admin-1992.pdf [https://perma.cc/4ATT-MYGU] ("ACIS was created . . . to support policies and principles of intellectual property law providing for a careful balance between the goals of strong protection and rewards for innovation, and the goals of interoperability, fair competition and open systems.").

71. *See* Attachment to Letter from Peter M.C. Choy to Professor Barry E. Carter (Nov. 5, 1992), https://www.ccianet.org/wp-content/uploads/2014/10/ACIS-Letter-to-Clinton-Admin-1992.pdf [https://perma.cc/5N3K-DSRB].

72. *See id.*

73. *See infra* text accompanying notes 370–73.

on network effects.[74] The war played out across various markets — from microcomputer operating systems to job scheduling software for mainframe computers, mobile phone networks, user interfaces, video game devices, printer cartridges, garage door openers, and all manner of application programs (business systems, design programs, video games, and spreadsheets). As the discussion below demonstrates, nearly every major software copyright litigation involved interoperability elements. Controlling the access features of software platforms produced the large-scale profits that could justify the costs of federal copyright litigation.

The courts faced daunting challenges in applying a complex new statute to a rapidly developing, technologically complex industry. Perhaps not surprisingly, they initially struggled to find the right balance. The Third Circuit's software copyright decisions in the mid to late 1980s put software copyright protection on a perilous path that threatened software innovation and competition. As I wrote in 1998, "[o]ver the course of the [next] decade, the federal courts [] reasserted fundamental limitations on the scope of copyright, effectively excluding network features from the domain of copyright protection."[75] I attributed the dramatic turnaround to copyright's adaptability to technological change, scholars' education of the courts about software technology, network economics, and the interplay of copyright and patent protection, and the federal judiciary's ability "to correct false starts and further the purposes . . . of copyright law within the broader framework of our intellectual property system."[76]

Unfortunately, it seems as if we are now at risk of repeating the mistakes of the 1980s. To understand the confusion that has emerged in the contemporary wave of API copyright litigation, it will be useful to trace the historical development of software copyright jurisprudence, as well as subsequent developments in copyright legislation.

1. Jurisprudence

The aphorism "bad facts make bad law"[77] captures the early development of software copyright jurisprudence. Such cases produced an inauspicious start to software copyright jurisprudence. But by the early 1990s, courts came to better appreciate both the technical as-

74. *See generally* Menell, *supra* note 12.

75. *Id.* at 652.

76. *Id.* at 653–54.

77. *See, e.g.*, Haig v. Agee, 453 U.S. 280, 319 (1981) (Brennan, J., dissenting) ("bad facts make bad law"); *see also* N. Sec. Co. v. United States, 193 U.S. 197, 400 (1904) (Holmes, J., dissenting) ("Great cases, like hard cases, make bad law."); *cf.* Frederick Schauer, *Do Cases Make Bad Law?*, 73 U. CHI. L. REV. 883, 884 (2006) (arguing that the act of deciding cases itself under the common law makes bad law).

pects of computer programming and how such works fit within copyright law.

i. The Early Years

The first major cases to address copyright protection for interoperable features of computer software pitted Apple Computer Corporation, then a young, break-out microcomputer company, against cavalier, unscrupulous competitors offering discount "interoperable" Apple clones.[78] The clone makers quickly entered the market by simply copying, bit by bit, Apple's operating system and application programs. In one case, the competitor had the audacity to call their competing computer system "Pineapple."[79] Not only did these companies not write the computer programs, they also did not even know what was in the source code. That enabled Apple to prove factual copying by pointing out a suspicious similarity between Franklin Computer's code and Apple's original code: the names of Apple programmers in a comment field.[80]

The defendants in these cases argued that copyright protection did not extend to non-human readable (object code[81]) formats of computer software and that the idea-expression doctrine barred copyright protection for operating systems. They further argued that copyright protection should not stand in the way of their selling computers that can run programs written for the Apple II.

Given the hard work that Apple put into developing the Apple II computer system and the bundled operating system and application programs, the courts had little trouble validating Apple's complaint that verbatim copying of millions of bits of code constituted copyright infringement. The 1976 Act, in conjunction with the CONTU Report,

78. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 545 F. Supp. 812 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983); Apple Comput.., Inc. v. Formula Int'l, Inc., 562 F. Supp. 775 (C.D. Cal. 1983), *aff'd*, 725 F.2d 521 (9th Cir. 1984).

79. Apple Comput., Inc. v. Formula Int'l, Inc., 562 F. Supp. at 777, 785 (C.D. Cal. 1983), *aff'd*, 725 F.2d at 526.

80. Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d at 1245.

81. Computers manipulate data according to a set of instructions called a computer program. At their most basic level, computer programs represent information and instruct computer devices through binary information ("0" (usually connoting "off") and "1" (usually connoting "on")). Strings of binary information can represent alphanumerical symbols, words, and images. Computer programs are typically written in high level, human-readable languages such as Fortran, C, and Java. Such "source code" programs are compiled using particular lexical, syntactic, and semantic rules into computer-readable "object code" for execution on a particular computer operating system. Programs written in high level, human-readable computer languages ("source code") are compiled into computer-readable "object code."

clearly extended copyright protection in this circumstance.[82] In that sense, the cases were easy.

Yet, due to the "bad facts" of blatant and cavalier piracy,[83] the Third Circuit went overboard in some of its dicta. In addressing the defendant's interoperability argument, the court opined that "total compatibility with independently developed application programs . . . is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[84] However, since two entirely different programs can achieve the same "certain result[s]"[85] — for example, generate the same set of protocols needed for interoperability — the court was not justified in making such an expansive statement about the scope of copyright protection for computer program elements. CONTU was clear that "[o]ne is always free to make the machine do the same thing as it would if it had the copyrighted work placed in it, but only by one's own creative effort rather than by piracy."[86] Given the verbatim copying of millions of bits of object code, there was no need to address the interoperability issue. The defendant failed to explain which elements of the program were protectable and which were not.

The next major software copyright appellate decision also arose in the Third Circuit. The bad facts in this case involved a messy consulting arrangement. In *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,[87] the owner of a dental laboratory hired a custom software firm to develop a computer program that would organize the bookkeeping and administrative tasks of its business. Whelan, the principal programmer, interviewed employees about the operation of the laboratory and then developed a program to run on the laboratory's IBM Series One computer. Under the terms of their agreement, Whelan retained the copyright in the program and agreed to use its best efforts to improve the program while Jaslow Laboratory agreed to use its best efforts to market the program. Rand Jaslow, an officer and shareholder of the laboratory, then created a version of the pro-

---

82. *See* Note, *Copyright Protection of Computer Object Code*, 96 HARV. L. REV. 1723, 1743–44 (1983). The emulation of particular aspects of a computer program, such as input formats, however, raised more complex API issues. *See, e.g.*, Synercom Tech., Inc. v. Univ. Computing Co., 462 F. Supp. 1003, 1011–12 (N.D. Tex. 1978).

83. After reporting that "Apple estimated the 'works in suit' took 46 man-months to produce at a cost of over $740,000, not including the time or cost of creating or acquiring earlier versions of the programs or the expense of marketing the programs," the Third Circuit noted that Franklin's vice-president of engineering "admitted copying each of the works in suit from the Apple programs" because "it was not feasible for Franklin to write its own operating system programs." *Apple*, 714 F.2d at 1245.

84. *See id.* at 1253.

85. *See* CONTU REPORT, at 12, 20.

86. *See id.* at 21.

87. 797 F.2d 1222 (3d Cir. 1986).

gram that would run on other computer systems. Whelan sued for copyright infringement.

At trial, the evidence showed that the Jaslow program did not literally copy Whelan's code, but there were overall structural similarities between the two programs. As a means of distinguishing protectable expression from unprotectable idea, the court reasoned:

> *[T]he purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea.* Where there are many means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.[88]

In applying this rule, the court defined the idea as "the efficient management of a dental laboratory," for which countless ways of expressing the idea would be possible.[89] Drawing the idea-expression dichotomy at such a high level of abstraction implies an expansive scope of copyright protection if all implementations of the idea constitute protectable expression. Furthermore, the court's conflation of merger analysis and the idea-expression dichotomy implicitly allows copyright protection of procedures, processes, systems, and methods of operation that are expressly excluded under § 102(b).[90]

Although the case did not directly address copyright protection for interoperable features of computer code, the court's mode of analysis expanded the scope of copyright protection for all aspects of computer programs. If everything below the general purpose of the program was protectable under copyright law, then it would follow that particular protocols were protectable because there would be other ways of serving the same general purpose of the program. Such a result would effectively bar competitors from developing interoperable programs and computer systems.

The next appellate decision to address the scope of protection for computer software also involved "bad facts": the "rogue employee"

---

88. *Id.* at 1236 (emphasis in original; citations omitted).

89. *Id.*

90. Lawyers representing plaintiffs in the early major cases embraced the *Whelan* decision. They analogized computer software to literary and dramatic works. *See* Anthony L. Clapes, Patrick Lynch & Mark R. Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 U.C.L.A. L. REV. 1493 (1987) (counsel for IBM and Lotus); Arthur Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977 (1993) (counsel for Lotus); Jack Brown, *'Analytical Dissection' of Copyrighted Computer Software-Complicating the Simple and Confounding the Complex*, 25 ARIZ. ST. L.J. 801 (1993) (counsel for Apple Computer Corp.)

scenario.[91] Johnson Controls had developed automated process control systems for wastewater treatment plants. Several of its former employees who were intimately familiar with this software formed Phoenix Control Systems, a competing company offering similar software products and services. After Johnson Controls sued for copyright infringement, misappropriation of trade secrets, unfair competition, trade libel, and interference with contractual relations, the district court granted a preliminary injunction prohibiting Phoenix Control Systems from copying, distributing, preparing derivatives of, publishing, or representing that they had the ability to use Johnson Controls' computer software.

Based on a detailed special master report identifying various similarities between the parties' programs, the district court concluded that there was ample basis for finding substantial similarity with Johnson Controls's protected expression.[92] In affirming the grant of the preliminary injunction, the Ninth Circuit explained that "[w]hether the nonliteral components of a program, including the structure, sequence and organization and user interface, are protected depends on whether, on the particular facts of each case, the component in question qualifies as an expression of an idea, or an idea itself."[93] The court's terse analysis notes the sophistication of Johnson Controls' program and comments that the creativity in the structure of the program "will no doubt be revisited at trial."[94] The decision does not refer to interoperability or APIs. It concludes merely that "[n]onliteral components of computer software may be protected by copyright where they constitute expression, rather than ideas."[95] The decision neither cites the *Whelan* case, which was decided more than two years prior to the Ninth Circuit argument, nor adopts its expansive analytic framework.

ii. The Modern Software Copyright Era

The *Whelan* idea/expression test was roundly criticized by commentators,[96] and other courts began developing alternative approaches to the scope of copyright protection that better comported with the fundamental principles of copyright protection. A few months after

91. *See* Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173 (9th Cir. 1989).

92. *Id.* at 1175–76.

93. *Id.* at 1175.

94. *Id.* at 1176.

95. *Id.* at 1177.

96. *See, e.g.*, *LaST Frontier Software Report*, *supra* note 5, at 20–21; Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, *supra* note 4, at 1074; John Englund, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 881 (1990); 4 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03(F)(1) (2017).

the *Whelan* decision, the Fifth Circuit confronted a similar claim of copyright infringement based upon structural similarities between two programs designed to provide cotton growers with accounting services, information regarding cotton prices and availability, and a means for conducting cotton transactions electronically.[97] In declining to follow the *Whelan* approach, the court found that the similarities in the programs were dictated largely by standard practices in the cotton market — what the court called "externalities" — such as the "cotton recap sheet" for summarizing basic transactional information. These externalities constituted unprotectable ideas.[98]

In 1992, the Second Circuit adapted Judge Learned Hand's seminal abstraction-filtration-comparison[99] test to computer software analysis.[100] Like many of the early software copyright cases, *Computer Associates v. Altai* again involved the rogue employee scenario. But unlike the Third Circuit in *Franklin* and *Whelan*, the Second Circuit focused on the foundational principles undergirding the intellectual property system and avoided loose and expansive dicta.

Computer Associates ("CA"), a leading mainframe software provider, had developed SCHEDULER, a job scheduling program[101] that worked with IBM mainframe computers. Part of the success of this program was that it had a sub-component, called ADAPTER, which interoperated with any of the three IBM mainframes (DOS/VSE, MVS, and VM/CMS). As a result, the user did not need to customize her programs for each of the IBM mainframes. ADAPTER ensured that programs written for SCHEDULER would run on any of the three IBM mainframes.

Altai was developing its own job scheduling software for the IBM mainframes, called OSCAR. It hired Claude Arney, a former CA programmer. Unbeknownst to Altai's management, Arney copied thirty percent of OSCAR's code from CA's ADAPTER program into Altai's ZEKE program.[102] When Altai management learned of the copying, the company initiated a clean room[103] rewrite of the program. Altai accepted responsibility for copyright infringement based on Arney's misdeeds and was ordered to pay $364,444 in damages.[104]

---

97. Plains Cotton Coop. Assoc. v. Goodpasture Comput. Serv., Inc., 807 F.2d 1256 (5th Cir. 1987).

98. *Id.* at 1262. The court found persuasive the decision in Synercom Tech., Inc. v. Univ. Computing Co., 462 F. Supp. 1003, 1013 (N.D. Tex. 1978), which analogized the "input formats" of a computer program (the organization and configuration of information to be inputted into a computer) to the "figure-H" pattern of an automobile stick shift.

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

100. *See* Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

101. *See Job Scheduler*, WIKIPEDIA, https://en.wikipedia.org/wiki/Job_scheduler [https://perma.cc/T8MG-EG6G].

102. *Altai*, 982 F.2d at 699.

103. *See supra* note 56 (defining clean room).

104. *Altai*, 982 F.2d at 696.

Altai did not challenge this ruling, but sought to market the revised clean room version of OSCAR. CA claimed that this version was also infringing due to structural similarities at various levels, such as flow charts, inter-modular relationships, parameter lists, and macros. The district court criticized *Whelan*'s "simplistic test" for determining similarity between computer programs.[105] It rejected the notion that there is but one idea per program and that as long as there were alternative ways of expressing that one idea, copyright law protected any particular version. Focusing on the various levels of the computer programs at issue, the court determined that the similarities between the programs were dictated by external factors — such as the interface specifications of the IBM operating system and the demands of functionality — and hence no protected code was infringed.[106]

On appeal, the Second Circuit fleshed out a detailed analytical framework for determining copyright infringement of computer code:

> In ascertaining substantial similarity . . . a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or perhaps kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program.[107]

The court's abstraction-filtration-comparison test recognized that an idea could exist at multiple levels of a computer program and not solely at the most abstract level. Furthermore, it set the ultimate comparison not between the programs as a whole, but between the *protectable* elements of the plaintiff's program and the allegedly infringing program. Of most importance with regard to fostering interoperability, the court held that copyright protection did not extend to those program elements where the programmer's "freedom to choose" is:

> [C]ircumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2)

105. Comput.. Assocs. Int'l v. Altai, Inc., 775 F. Supp. 544, 558 (E.D.N.Y. 1991).
106. *Id.* at 558–62.
107. *Altai*, 982 F.2d at 706.

> compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.[108]

Directly rejecting the dictum in *Apple v. Franklin*,[109] the Second Circuit recognized that external factors such as interface specifications, de facto industry standards, and accepted programming practices are not protectable under copyright law. The formulation of the Second Circuit test judges these external factors at the time of the allegedly infringing activities (that is, ex post), not at the time that the first program is written.[110]

Commentators warmly embraced the *Altai* decision,[111] and the abstraction-filtration-comparison approach has been universally adopted by the courts.[112]

The *Altai* case addressed programmers' freedom to write code to interoperate with APIs established by a third party: in that case, by IBM. IBM had not challenged either CA's or Altai's use of its interface specifications. It welcomed other companies developing software

---

108. *Id.* at 709–10. The court observed that "[w]hile, hypothetically, there might be a myriad [sic] ways in which a programmer may effectuate certain functions within a program — *i.e.*, express the idea embodies in a given subroutine — efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable operations." *Id.* at 708.

109. *See* Apple Comput. v. Franklin Comput. Corp., 714 F.2d 1240, 1244. (3d Cir. 1983).

110. The court emphasized that the first to write a program for a particular application should not be able to "'lock up' basic programming techniques as implemented in programs to perform particular tasks." 982 F.2d at 712 (quoting Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, *supra* note 4, at 1087).

111. *See* David Bender, Computer Associates v. Altai: *Rationality Prevails*, 9(8) THE COMPUTER LAWYER 1 (Aug. 1992); Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, *supra* note 4, at 2652; Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1 (1995).

112. *See* Menell, *supra* note 22, at 84–85; Lemley, *supra* note 111 (collecting cases). In *Gates Rubber v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 836–43 (10th Cir. 1993), the Tenth Circuit expressly expanded the range of external factors to be used in filtering out unprotectable elements to include hardware standards and mechanical specifications, software standards and compatibility requirements, industry programming practices, and practices and demands of the industry being serviced. The court also noted that processes used in designing a computer system, or components therein (e.g., modules, algorithms), must also be filtered out as unprotectable under § 102(b). While not ruling that interface specifications are uncopyrightable as a matter of law, the Eleventh Circuit's decision in *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1547 (11th Cir. 1996), held that "external considerations such as compatibility may negate a finding of infringement." The court commented that "[i]t is particularly important to exclude methods of operation and processes from the scope of copyright in computer programs because much of the content of computer programs is patentable. Were we to permit an author to claim copyright protection for those elements of the work that should be the province of patent law, we would be undermining the competitive principles that are fundamental to the patent system." *Id.* at 1542 n.21.

for its mainframes. Thus, the case did not specifically address whether an API developer could assert a copyright infringement claim based on unauthorized use of its own interface specifications. That issue would emerge in a series of cases involving video games and spreadsheets.

The "bad facts" pattern continued in *Atari Games Corp. v. Nintendo of America*,[113] an early video game interoperability case. Nintendo embedded software security code in a patented computer chip on its entertainment console and authorized game cartridges. Nintendo kept the lock-out code secure by distributing it only on computer chips. Thus, the code was embedded in microprocessor chip layers that could not be readily decrypted. Atari Games sought to decrypt that code so that it could sell video games for the Nintendo game console without having to license the proprietary chip. After failing to hack the chip, Atari Games gained access to Nintendo's source code from the Copyright Office based on a misleading assertion that it was facing actual or prospective litigation.[114] With the source code in hand and in violation of Copyright Office regulations,[115] Atari Games deciphered the lock-out code and developed an interoperable program. After finding that Atari Games copied "more [computer code] than was needed to make a game work on the [Nintendo] console," the district court granted a preliminary injunction enjoining Atari Games from manufacturing or distributing Nintendo's computer program.[116]

Atari Games appealed the decision to the Federal Circuit.[117] Applying Ninth Circuit law, the Federal Circuit affirmed the grant of the preliminary injunction. The court further explained that:

> Nintendo seeks to protect the creative element of its program beyond the literal expression used to effect the unlocking process. The district court defined the unprotectable . . . idea or process as the generation of a data stream to unlock a console. This court discerns no clear error in the district court's conclusion. The

113. 18 U.S.P.Q.2d 1935 (N.D. Cal. 1991), *aff'd*, 975 F.2d 832 (Fed. Cir. 1992).

114. *See Atari*, 975 F.2d at 841.

115. Requesters agree "not to copy . . . the material to be inspected." *See* U.S. COPYRIGHT OFFICE, COMPENDIUM OF COPYRIGHT OFFICE PRACTICES II § 1902.01, http://www.copyright.gov/history/comp/compendium-two-1988.pdf [https://perma.cc/7P3T-CXGL]; *see also* 37 U.S.C. § 201.2(d)(2) (as amended through July 1, 1986) (permitting "reproduction only if: (1) the copyright owner grants permission, (2) a court orders reproduction, or (3) . . . (ii) The Copyright Office receives a written request from an attorney on behalf of either the plaintiff or defendant in connection with litigation, actual or prospective, involving the copyrighted work . . . .").

116. *Atari*, 18 U.S.P.Q.2d at 1940.

117. The patent infringement claims in the case vested exclusive appellate jurisdiction with the Federal Circuit. *See* Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990).

> unique arrangement of computer program expression which generates that data stream does not merge with the process so long as alternate expressions are available. *Formula Int'l*, 725 F.2d at 525. In this case, Nintendo has produced expert testimony showing a multitude of different ways to generate a data stream which unlocks the [Nintendo] console.[118]

The Federal Circuit implies that Atari Games could have avoided copyright infringement had it gained access to the lock-out code legitimately and independently written the implementing code: "[w]hen the nature of a work requires intermediate copying to understand the ideas and processes in a copyrighted work, that nature supports a fair use for intermediate copying. Thus, reverse engineering object code to discern the unprotectable ideas in a computer program is a fair use."[119] The clear implication is that the particular lock-out code is an unprotectable idea, because there is no other expression that achieves the same function. Nonetheless, the court rejected Atari Games' fair use defense because Atari Games procured Nintendo's source code unlawfully.[120] The court further chastised Atari Games for replicating more computer code from the unlock chip in its game cartridges than was necessary to accomplish the unlock function.[121]

---

118. *Atari*, 975 F.2d at 840.

119. The Federal Circuit emphasized the principle that the fair use doctrine generally "permits an individual in rightful possession of a copy of a work to undertake necessary efforts to understand the work's ideas, processes, and methods of operation." *Id.* at 842. The court noted that "[a]n author cannot acquire patent-like protection by putting an idea, process, or method of operation in an unintelligible format and asserting copyright infringement against those who try to understand that idea, process, or method of operation." *Id.* Applying these principles, the court reasoned that "[w]hen the nature of a work requires intermediate copying to understand the ideas and processes in a copyrighted work, that nature supports a fair use for intermediate copying. Thus, reverse engineering object code to discern the unprotectable ideas in a computer program is a fair use." *Id.* at 843.

120. *Id.* at 841–44 ("To invoke the fair use exception, an individual must possess an *authorized copy* of a literary work." (emphasis added)).

121. *Id.* at 843–45 ("Any reproduction of protectable expression must be strictly necessary to ascertain the bounds of protected information within the work."). The court notes that:

> Nintendo modified its . . . chip program in 1987. This modification deleted some instructions from the original [] program. Nonetheless the [Atari Games] program contains instructions equivalent to those deleted from the original [Nintendo] program. These unnecessary instructions strongly suggest that the [Atari Games] program is substantially similar to the [Nintendo] program. *See, e.g.*, *M. Kramer Mfg. Co. v. Andrews*, 783 F.2d 421, 446 (4th Cir. 1986) ("Courts have consistently viewed 'common errors' as strongest evidence of copying.") . . . .

Id. at 845. This passage indicates that the Federal Circuit conflated factual copying (which focuses on probative similarity) with legal copying (which focuses on substantial similarity of protected expression). *See* Johnson v. Gordon, 409 F.3d 12 (1st Cir. 2005); NIMMER ON COPYRIGHT, *supra* note 96, at § 13.03(A) (explicating the distinction between probative and

The Ninth Circuit's decision later that year in *Sega Enterprises Ltd. v. Accolade*[122] expressly recognized the legitimacy of deciphering and copying particular lock-out codes for purposes of developing interoperable products. Like Nintendo, Sega developed a successful video game platform called Genesis for which it licensed access to video game developers. Accolade, a video game manufacturer, wanted to distribute versions of its game on the Genesis platform. It did not, however, want to limit distribution exclusively to Genesis, as Sega required. Rather than license access to Sega's code, Accolade reverse engineered the access code through a painstaking effort that entailed making hundreds of intermediate copies of Sega's computer code. Accolade then incorporated only those code elements that were necessary to achieve interoperability with the Genesis platform into Accolade game cartridges.[123] Ultimately, the amount copied was only about 25 bytes, placed into games containing between 500,000 and 1,500,000 bytes.[124]

Sega sued Accolade for copyright and trademark infringement.[125] In view of the relatively small amount of Sega code in the Accolade game cartridges, Sega focused its copyright claim on the making of intermediate copies of its full computer program during the reverse engineering process. The district court rejected Accolade's argument that such intermediate copies constituted fair use and granted a preliminary injunction.[126]

The Ninth Circuit reversed the district court decision, holding that "disassembly of object code in order to gain an understanding of the ideas and functional concepts embodied in the code is a fair use that is privileged by section 107 of the Act."[127] The court determined that the policies underlying the Copyright Act authorize disassembly of copyrighted object code and the making of intermediate copies to identify elements of code that are not protected by copyright law.[128] In reaching this conclusion, the Ninth Circuit ruled that the "functional requirements for compatibility with the Genesis [video game console

---

substantial similarity). In any case, without seeing how much code was copied into the Atari Games' video games, it is not possible to assess the Federal Circuit's assertion that Atari Games' copying of Nintendo code constituted substantial similarity of protected expression.

122. 977 F.2d 1510 (9th Cir. 1993).

123. *Id.* at 1516.

124. *See id.*

125. The basis for the trademark claim was that the initialization code prompted a visual display for approximately three seconds that read "PRODUCED BY OR UNDER LICENSE FROM SEGA ENTERPRISES LTD." *Id.* at 1515–16.

126. *See* Sega Enters. v. Accolade, Inc., 785 F. Supp. 1392, 1397–1400 (N.D. Cal. 1992), *rev'd*, 977 F.2d 1510 (9th Cir. 1993).

127. *Sega*, 977 F.2d at 1517.

128. *See id.*

are] aspects of Sega's programs that are not protected by copyright. 17 U.S.C. § 102(b)."[129]

In discussing the nature of the copyrighted work, the second fair use factor, the Ninth Circuit addressed the application of the idea-expression dichotomy to computer code. The court rejected the *Whelan* approach as "simplistic and overbroad" and endorsed the *Altai* approach as the appropriate framework.[130] "Under a test that breaks down a computer program into its component subroutines and sub-subroutines and then identifies the idea or core functional element of each, such as the test recently adopted by the Second Circuit in [*Altai*], many aspects of the program are not protected by copyright."[131] In explaining why disassembly and reproduction of object code constitutes fair use, the court held that the "functional specifications" of a computer program are unprotectable.[132] In *Sega*, such specifications operated the lock-out functionality. Thus, the court held that the particular code or process for interoperating with a copyrighted computer program was not protected by copyright law.[133]

The Ninth Circuit based its analysis on the architecture of the intellectual property system:

> If disassembly of copyrighted object code is *per se* an unfair use, the owner of the copyright gains a *de facto* monopoly over the functional aspects of his work — aspects that were expressly denied copyright protection by Congress. 17 U.S.C. § 102(b). In order to enjoy a lawful monopoly over the idea or functional principle underlying a work, the creator of the work must satisfy the more stringent standards imposed by the patent laws. *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 489 U.S. 141, 159–64 (1989). Sega does not hold a patent on the Genesis console.[134]

The Ninth Circuit reaffirmed and expanded the *Sega* analysis in *Sony Computer Entertainment, Inc. v. Connectix Corp.*,[135] further ce-

129. *Id.* at 1522.
130. *See id.* at 1524–25.
131. *See id.* at 1525.
132. *See id.* at 1526.
133. The court notes that its fair use analysis "does not, of course, insulate Accolade from a claim of copyright infringement with respect to its finished products. Sega has reserved the right to raise such a claim, and it may do so on remand." *See id.* at 1528. The fact that Accolade copied only 25 bytes of code needed for interoperability explains why the issue was never pursued.
134. *See id.*
135. 203 F.3d 596 (9th Cir. 2000).

menting the foundational premise that copying code and processes necessary for interoperability does not constitute copyright infringement.

The Northern District of California and the Ninth Circuit applied the *Altai* framework to the graphical user interface features of a computer program in *Apple Computer, Inc. v. Microsoft Corp*.[136] Apple alleged that Microsoft's Windows operating system infringed Apple's copyrights in the desktop graphical user interface for its Macintosh computer system. The copyright issue was muddied by the existence of a licensing agreement authorizing the defendants to use aspects of Apple's graphical user interface. The court determined, however, that the licensing agreement was not a complete defense to the copyright claims and therefore undertook an analysis of the scope of copyright protection for a large range of audiovisual elements of computer screen displays.[137]

In framing the analysis, the district court expressly recognized the relevance of network externalities and the cumulative nature of innovation to the scope of copyright protection:

> Copyright's purpose is to overcome the public goods externality resulting from the non-excludability of copier/free riders who do not pay the costs of creation. Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1059 (1989). But overly inclusive copyright protection can produce its own negative effects by inhibiting the adoption of compatible standards (and reducing so-called 'network externalities'). Such standards in a graphical user interface would enlarge the market for computers by making it easier to learn how to use them. *Id.* at 1067–70. Striking the balance between these considerations, especially in a new and rapidly changing medium such as computer screen displays, represents a most ambitious enterprise. *Cf. Lotus Dev. Corp. v. Paperback Software Int'l*, 740 F. Supp. 37 (D. Mass. 1990).
>
> While the Macintosh interface may be the fruit of considerable effort by its designers, its success is the result of a host of factors, including the decision

136. 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.3d 1435 (9th Cir. 1994).

137. *See* Apple Comput., Inc. v. Microsoft Corp., 709 F. Supp. 925, 930 (N.D. Cal. 1989); Apple Comput., Inc. v. Microsoft Corp., 717 F. Supp. 1428 (N.D. Cal. 1989); Apple Comput., Inc. v. Microsoft Corp., 759 F. Supp. 1444 (N.D. Cal. 1991).

> to use the Motorola 68000 microprocessor, the tactical decision to require uniform application interfaces, and the Macintosh's notable advertising. And even were Apple to isolate that part of its interface's success owing to its design efforts, lengthy and concerted effort alone 'does not always result in inherently protectible [sic] expression.' [quoting *Computer Associates v. Altai*, 982 F.2d at 711.]
>
> By virtue of having been the first commercially successful programmer to put these generalized features together, Apple had several years of market dominance in graphical user interfaces until Microsoft introduced Windows 3.0, the first DOS-based windowing program to begin to rival the graphical capability of the Macintosh . . . . To accept Apple's 'desktop metaphor'/'look and feel' arguments would allow it to sweep within its proprietary embrace not only Windows and NewWave but, at its option, also other desktop graphical user interfaces which employ the standardized features of such interfaces, and to do this without subjecting Apple's claims of copyright to the scrutiny which courts have historically employed. Apple's copyrights would hold for programs in existence now or in the future — for decades. One need not profess to know for sure where should lie the line between expression and idea, between protection and competition to sense with confidence that this would afford too much protection and yield too little competition.
>
> The importance of such competition, and thus improvements or extensions of past expressions, should not be minimized. The Ninth Circuit has long shown concern about the uneasy balance which copyright seeks to strike:
>
> > What is basically at stake is the extent of the copyright owner's monopoly — from how large an area of activity did Congress intend to allow the copyright owner to exclude others?[138]

138. *Apple*, 799 F. Supp. at 1025–26 (quoting Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971)).

The court found that all of the alleged similarities between Apple's works and Windows not authorized by the licensing agreement were either not protectable or subject to at least one of the limiting doctrines.[139] As a result, the court applied the "virtual identity" standard in comparing the works as a whole[140] and determined that no infringement had occurred.[141] On appeal, the Ninth Circuit affirmed the district court's dissection of the work in question to determine which elements were protectable, its filtering out of unprotectable elements, and its application of the "virtual identity" standard in this context.[142]

The copyrightability of command systems for computer software arose most directly in litigation surrounding spreadsheet technology. Building upon the success of the VisiCalc program developed for the Apple II computer, Lotus Corporation marketed an enhanced and faster operating spreadsheet program incorporating many of VisiCalc's features and commands into its 1-2-3 program for the IBM PC platform. Lotus 1-2-3 quickly became the market leader for spreadsheets running on IBM and IBM-compatible machines, and knowledge of the program became a valuable employment skill in the accounting and management fields. As illustrated in Figure 1, the 1-2-3 command hierarchy was particularly attractive because it provided a logical structuring of more than two hundred commands. It also enabled users to automate particular accounting and business planning functions with customized programs called macros. Businesses and users increasingly became "locked in" to the 1-2-3 command structure as their human capital investments in learning the system and library of macros grew.[143] By the late 1980s, software developers seeking to enter the spreadsheet market could not ignore the large premiums that many consumers placed on being able to use their investments in the 1-2-3 system in a new spreadsheet environment, even when a new spreadsheet product offered significant technological improvements over the Lotus spreadsheet.[144]

---

139. *See id.* at 1025–42.

140. The Ninth Circuit developed the heightened "virtual identity" standard for evaluating thinly protected works such as compilations of simple, narrowly protected elements, such as the visual layout of a day planner (comprising a calendar and ruled lines), *see Harper House, Inc. v. Thomas Nelson, Inc.*, 889 F.2d 197 (9th Cir. 1989), and the audiovisual elements for a karate videogame, *Data East USA, Inc. v. Epyx, Inc.*, 862 F.2d 204 (9th Cir. 1988).

141. *See* 799 F. Supp. at 1042–47.

142. Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994).

143. *See* Neil Gandal, *Hedonic Price Indexes for Spreadsheets and an Empirical Test for Network Externalities*, 25 RAND J. ECON. 160 (1994).

144. *See* Mike Hogan, *Product Outlook: Fresh from the Spreadsheet Oven*, PC WORLD, Feb. 1988, at 100–02; Lawrence J. Magid, *'Surpass' Spreadsheet Program Lives Up to Name, Beats Lotus 1-2-3*, WASH. POST, Apr. 25, 1988, at 26.

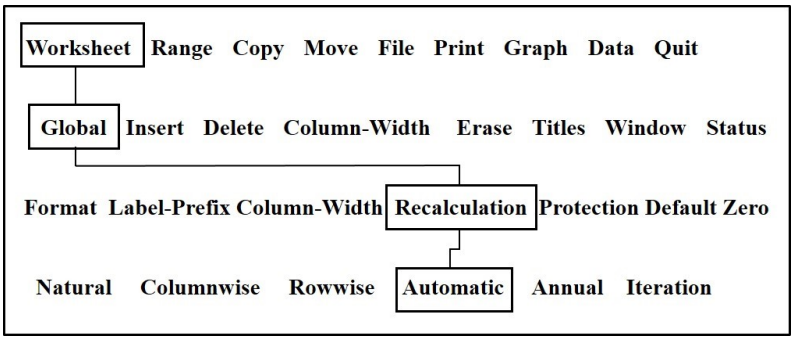

Figure 1. Lotus 1-2-3 Menu Command Hierarchy.

In the mid-1980s, Paperback Software International introduced a spreadsheet program called VP-Planner that largely emulated the operation of the Lotus 1-2-3 product.[145] Paperback was careful to ensure that the program code did not copy the 1-2-3 source or object code. Nonetheless, Lotus sued Paperback for copyright infringement, alleging that VP-Planner inappropriately copied the 1-2-3 menu structure, which included the choice of command terms, the structure and order of those terms, their presentation on the screen, and the long prompts. Relying on the Third Circuit's *Whelan* framework and hence focusing simply upon whether such elements could be expressed in a variety of ways, Judge Keeton of the District of Massachusetts found for Lotus. Facing bankruptcy, Paperback agreed not to appeal the judgment as part of a settlement.[146]

After three years of intensive development efforts, Borland International, developer of several successful software products including Turbo Pascal and Sidekick, introduced Quattro Pro, its entry into the spreadsheet market. Unlike Paperback's VP-Planner spreadsheet, which offered little beyond the 1-2-3 product, Quattro Pro made substantial design and operational improvements and earned accolades in the computer product review magazines.[147] Also unlike VP-Planner, Quattro Pro offered a new interface for its users, which many purchasers of spreadsheets preferred over the 1-2-3 interface. Nonethe-

145. *See* Tracy R. Licklider, *Ten Years of Rows and Columns*, BYTE, Dec. 1989, at 324.

146. *See* Andrew Ould, *Legal Dispute Kept Paperback from Lotus Appeal*, PC WEEK, Jan. 21, 1991, at 138.

147. *See Spreadsheet, Borland International Inc.'s Quattro Pro for Windows and Quattro Pro 4.0 for DOS*, PC-COMPUTING, Dec. 1992, at 140 ("No doubt about it: Quattro Pro for DOS is the best DOS spreadsheet there is. Period."); *Borland's Quattro Pro Tops 2.5 Million Units Shipped*, BUS. WIRE, Jul. 1, 1992 ("Since its introduction in October 1989, Quattro Pro has won an unprecedented 42 industry awards and honors worldwide from users and product reviewers."); *Software Review, Quattro Pro 4.0; Borland International Inc.'s Spreadsheet Software*, COMPUTER SHOPPER, Jun. 1992, at 536 ("Quattro Pro 4.0 simply shames other DOS-based spreadsheets, especially Lotus 1-2-3 r2.").

less, because of the large number of users who were already familiar with the 1-2-3 command structure and who had made substantial investments in developing macros to run on the 1-2-3 platform, Borland considered it essential to offer an operational mode based on the 1-2-3 command structure as well as macro compatibility. Unlike VP-Planner, Borland's visual representation of the 1-2-3 command mode substantially differed from the 1-2-3 screen displays.

To clarify the legal status of its product, Borland brought a declaratory judgment action in California. Through astute jurisdictional maneuvering, Lotus consolidated the Borland case with the Paperback case before Judge Keeton. After protracted litigation,[148] Judge Keeton found for Lotus. Following the *Whelan* framework, he held that a menu command structure was protectable if there were many such structures theoretically available. He also found that Borland was not permitted to achieve macro compatibility with the 1-2-3 product, distinguishing the treatment of external constraints noted in the *Altai* decision because such constraints had to exist when the first program was created. Thus, Judge Keeton effectively ruled that constraints governing the design of computer systems must be analyzed ex ante (based on technical considerations at the time the first program is written) and not ex post (after the market has operated to establish a de facto standard).

By the time Borland's appeal reached the First Circuit, the Second Circuit's *Altai* decision had received a favorable reception in professional and academic journals[149] and its approach had been adopted by several courts.[150] The Ninth Circuit and the Federal Circuit had issued the *Sega* and *Atari Games* decisions, further emphasizing the legitimacy of developing interoperable systems. In addition, the Supreme Court's decision in *Feist Publications, Inc. v. Rural Telephone Service Co.*,[151] denying copyright protection for alphabetically organized telephone directories for lack of originality, repudiated the "sweat of the brow" doctrine[152] and reaffirmed the "long recognized"

148. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 788 F. Supp. 78 (D. Mass. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203 (D. Mass. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 202 (D. Mass. 1993); Lotus Dev. Corp. v. Borland Int'l, Inc., 831 F. Supp. 223 (D. Mass. 1993).

149. *See* Bender, *supra* note 111, at 1; Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, *supra* note 4, at 2652; Lemley, *supra* note 111.

150. *See* Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1445 (9th Cir. 1994); Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1342–43 (5th Cir. 1994); Gates Rubber v. Bando Chem. Indus., Ltd., 9 F.3d 823, 841 (10th Cir. 1993).

151. 499 U.S. 340 (1991).

152. Several lower courts had found that copyright could be established on the basis of substantial effort in gathering facts. *See, e.g.*, Leon v. Pac. Tel. & Tel. Co., 91 F.2d 484 (9th Cir. 1937); Jeweler's Circular Publ'g Co. v. Keystone Publ'g Co., 281 F. 83 (2d Cir. 1922). The Supreme Court's *Feist* decision rejected this "sweat of the brow" theory in holding that originality is a requirement of copyright and therefore, unless a factual work exhibits originality as a compilation, it does not receive protection under the Copyright Act.

principle "that the fact/expression dichotomy limits severely the scope of protection in fact-based works."[153] Thus, systematic hierarchical frameworks based on mathematical and accounting systems, even though laboriously compiled, might not qualify for copyright protection. Furthermore, the *Borland* case had attracted tremendous interest among academics and interest groups skeptical of overbroad copyright protection for computer software.[154]

The First Circuit viewed the case as one of first impression: "[w]hether a computer menu command hierarchy constitutes copyrightable subject matter."[155] The court properly distinguished *Altai* as dealing with the protection of computer code as opposed to the results of such code.[156] Instead, the First Circuit saw the subject matter of the *Lotus* case as a "method of operation" falling directly within the exclusions from copyright protection set forth in § 102(b):

> We think that 'method of operation,' as that term is used in § 102(b), refers to the means by which a person operates something, whether it be a car, a food processor, or a computer. Thus a text describing how to operate something would not extend copyright protection to the method of operation itself; other people would be free to employ that method and to describe it in their own words. Similarly, if a new method of operation is used rather than described, other people would still be free to employ or describe that method.
>
> We hold that the Lotus menu command hierarchy is an uncopyrightable 'method of operation.' The Lotus menu command hierarchy provides the means by which users control and operate Lotus 1-2-3. If users wish to copy material, for example, they use the 'Copy' command. If users wish to print material, they use the 'Print' command. Users must use the command terms to tell the computer what to do. Without the menu command hierarchy, users would not be able to access and control, or indeed make use of, Lotus 1-2-3's functional capabilities.
>
> The Lotus menu command hierarchy does not merely explain and present Lotus 1-2-3's functional capabilities to the user; it also serves as the method

153. 499 U.S. at 350.

154. Amicus briefs were filed on behalf of computer scientists, intellectual property professors, and computer industry organizations.

155. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 813. (1st Cir. 1995).

156. *Id.* at 814.

> by which the program is operated and controlled . . . .[157]

The Supreme Court granted certiorari and affirmed without opinion by an equally divided vote.[158]

Subsequent appellate decisions reached similar outcomes, although they did not fully embrace the First Circuit's reasoning. In *MiTek Holdings, Inc. v. ARCE Engineering Co.*,[159] the holder of a copyright in an application program that designed and arranged wood trusses for framing roofs brought an infringement action against the maker of a competing program that featured a similar menu command tree and user interface. Affirming the lower court's decision, the Eleventh Circuit held that the menu and submenu command structure of the truss design program was uncopyrightable under § 102(b) because it represented a process.[160] The court did not need to reach the broader question, addressed in *Lotus*, of whether all menu command structures are uncopyrightable as a matter of law.

In *Mitel, Inc. v. Iqtel, Inc.*,[161] Mitel, the maker of a widely adopted computer system for automating the selection of a particular long-distance telephone carrier and remotely activating optional telecommunications features such as speed dialing, sued a competing firm that used identical command codes for copyright infringement. Because Mitel's system had become a de facto standard in the market, Iqtel defended its use of compatible controller codes on the ground that "technicians who install call controllers would be unwilling to learn Iqtel's new set of instructions in addition to the Mitel command code set, and the technician's employers would be unwilling to bear the cost of additional training."[162] Like Borland's Quattro, Iqtel's product included both its own command codes as well as a "Mitel Translation Mode." While commenting that a method of operation may in some circumstances contain copyrightable expression, the Tenth Circuit nonetheless concluded that Mitel's command codes, which were arbitrarily assigned, lacked the minimal degree of creativity necessary to qualify for copyright protection.[163] The court further held that Mitel's command codes should be denied copyright protection under the *scènes à faire* doctrine because they are largely dictated by external

157. *Id.* at 815.

158. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996) (Justice Stevens recused himself from participation in consideration of the case).

159. 89 F.3d 1548 (11th Cir. 1996).

160. *Id.* at 1556–57.

161. 124 F.3d 1366 (10th Cir. 1997).

162. *Id.* at 1369.

163. *Id.* at 1373–74.

factors such as hardware compatibility requirements and industry practices.[164]

Thus, although the Eleventh and Tenth Circuits did not expressly hold that all menu command hierarchies are uncopyrightable as a matter of law, the outcomes of *MiTek* and *Mitel* aligned with the First Circuit's holding in *Lotus*. There were no further reported cases addressing copyright protection for APIs over the next fifteen years.

2. Legislative Developments

The uncopyrightability of interoperable features of computer software arose as part of legislative deliberation over the passage of the DMCA.[165] Title I generally prohibits circumvention of technical protection measures put in place by copyright owners to protect copyrighted works.[166] Various interest groups advocated exempting circumvention for the purpose of developing interoperable computer programs and devices. Congress obliged by enacting § 1201(f)(1), which provides that:

> [A] person who has lawfully obtained the right to use a copy of a computer program may circumvent a technological measure that effectively controls access to a particular portion of that program for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs, and that have not previously been readily available to the person engaging in the circumvention, to the extent any such acts of identification and analysis do not constitute infringement under this title.[167]

The legislative history notes that this provision is:

> [I]ntended to allow legitimate software developers to continue engaging in certain activities for the purpose of achieving interoperability to the extent permitted by law prior to the enactment of this chapter. The objective is to ensure that the effect of current case law interpreting the Copyright Act is not

164. *Id.* at 1374–76.

165. Digital Millennium Copyright Act, Pub. L. 105-304, 112 Stat. 2860 (1998).

166. *See* WIPO Copyright and Performances and Phonograms Treaties Implementation Act, codified at 17 U.S.C. §§ 1201–05. (2012).

167. 17 U.S.C. § 1201(f) (2012).

> changed by enactment of this legislation for certain acts of identification and analysis done in respect of computer programs. *See*, *Sega Enterprises Ltd. v Accolade, Inc.*, 977 F.2d 1510, 24 U.S.P.Q.2d 1561 (9th Cir. 1992). The purpose of this section is to foster competition and innovation in the computer and software industry.[168]

Thus, in crafting the DMCA, Congress expressed its support for the *Sega* decision and recognized its importance for "foster[ing] competition and innovation in the computer and software industry."

D. The End of the First API Copyright War and the Logic of the Intellectual Property System

After an inauspicious start, the federal courts implemented a balanced framework for both protecting computer software against piracy and interpreting the idea-expression doctrine to ensure that copyright law excludes functional features of computer technology. These decisions have effectuated the subtle balance to which the CONTU Report referred.[169] The courts have come to appreciate that creativity must be understood contextually. While programming a computer can unquestionably be considered creative in a general sense, it might nonetheless be uncopyrightable due to functional characteristics. The design of an efficient mechanical machine likewise can be creative, but such devices are not eligible for copyright protection unless the aesthetic features can be separated from the functional attributes under the useful article doctrine.[170] Lines of code are the gears and levers of digital machines. The fact that computer software, like a sculptural work, is eligible for copyright protection does not authorize protection for functional features.[171]

The courts have come to recognize that APIs have significant functional dimensions. They serve in many contexts as the basis for interoperability of computer technologies and the particular functional specifications, as opposed to the implementing code, of a software program can be fairly characterized as "methods of operation." Although the Supreme Court's split decision in *Lotus v. Borland* left

---

168. *See* S. Rep. No. 105-190, at 13; (1998); *see also id.* at 32–34 (section-by-section analysis).

169. *See generally*, Menell, *supra* note 12, at 707–08.

170. *See* 17 U.S.C. § 101 (2012) ("Pictorial, graphic, and sculptural works" include two-dimensional and three-dimensional works . . .; the design of a useful article . . . shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article.").

171. *See* 17 U.S.C. § 102(b) (2012).

some uncertainty,[172] the resolution of that litigation marked the end of the major API copyright litigations that had raged since the early 1980s.

Precedential rulings in all courts of appeals applying copyright law's limiting doctrines to the functional elements of software rejected the *Apple v. Franklin* dictum that "total compatibility with independently developed application programs . . . is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[173] Courts outside of the Third Circuit also expressly rejected the *Whelan* framework for analyzing the structure, sequence, and organization of computer software. Congress expressly endorsed the *Sega* decision in adopting an interoperability exemption to the DMCA's anti-circumvention provisions. Furthermore, a unanimous Supreme Court decision in *TrafFix Devices, Inc. v. Marketing Displays, Inc.*[174] — which guarded against protection for functional features in trade dress — fortified the principle that utility patent law is the sole regime for protecting functional features and that courts should carefully guard against overprotection of intellectual works. By the turn of the millennium, the first API copyright war had come to an end.[175]

III. COPYRIGHT PROTECTION FOR COMPUTER SOFTWARE 2.0: THE *ORACLE* WAVE

Following the resolution of the first API copyright war, the software engineering community came to view high-level functions, labeling conventions, and the functional specifications of APIs as unprotectable under copyright law.[176] These norms were reinforced by

---

172. Notwithstanding the divided result, Justice Stevens likely would have sided with the First Circuit. He had generally taken less protectionist positions in intellectual property cases. *See, e.g.*, Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417 (1984) (limiting indirect copyright liability of device manufacturers); Parker v. Flook, 437 U.S. 584 (1978) (limiting patent protection for computer-related technologies).

173. Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253. (3rd Cir. 1983).

174. 532 U.S. 23 (2001).

175. *See* Menell, *supra* note 12.

176. *See* Brian Profitt, *The Impact of Oracle's Defense of API Copyrights*, ITWORLD (Aug. 23, 2011), http://www.itworld.com/article/2738675/mobile/the-impact-of-oracle-s-defense-of-api-copyrights.html [https://perma.cc/PVC9-2GWR] (observing that "[h]istorically, APIs have been regarded as not falling under copyright — the reasoning being that APIs are not creative implementations but rather statements of fact," but also noting the issue had been clouded by the distinction between "open" and "closed" APIs); *see also* Michael Hussey, *Copyright Captures APIs: A New Caution For Developers*, TECHCRUNCH (Nov. 3, 2015), https://techcrunch.com/2015/11/03/copyright-captures-apis-a-new-caution-for-developers/ [https://perma.cc/37XG-HE2Z] (observing that "[s]oftware developers routinely treat APIs as exempt from copyright protection"). *But see* Edward J.

the spread of open source software.[177] Furthermore, as the economics of network effects and interoperability suggests, many computer hardware and software companies actively sought platform adopters.[178] The Internet ushered in a new economic era in which companies could give away software and services while earning money from other sources, principally advertisers. Consistent with these patterns, Jonathan Schwartz, Sun's Chief Executive Officer, publicly congratulated Google on its decision to use Java software in Android,[179] proclaiming that Google had "strapped another set of rockets to the [Java] community's momentum — and to the vision defining opportunity across our (and other) planets."[180]

Thus, Oracle's filing of a lawsuit against Google over the Android platform's use of Java came as a surprise to many in the high technology community.[181] Yet to Sun and Google insiders, the writing was on the wall. Schwartz and his Sun colleagues were gravely concerned about Google's Android strategy at the time that Schwartz publicly celebrated the release of the Android Software Development Kit ("SDK").[182] Sun's hardware business had long been in decline and the company desperately needed to find ways to recoup its ongoing investments in Java. It actively pursued a strategy to establish its Java ME (Micro Edition) platform for embedded and mobile devices.[183] The congratulatory blog post was aimed at bringing Google to the

---

Naughton, *Copyright in APIs: The Sky Won't Fall, and The Clouds Are Safe*, EMERGING TECHNOLOGIES BLOG (May 30, 2012), http://brownrudnick.com/blog/emerging-technologies/copyright-in-apis-the-sky-wont-fall-and-the-clouds-are-safe/[https://perma.cc/5NHQ-64EZ] (questioning the validity of the "long-held practice of API copyright exemption").

177. *See generally* STEVEN WEBER, THE SUCCESS OF OPEN SOURCE (2005).

178. *See* Joe Mullin, *Sun's Jonathan Schwartz at Trial: Java Was Free, Android Had No Licensing Problem*, ARS TECHNICA (May 11, 2016), https://arstechnica.com/tech-policy/2016/05/suns-jonathan-schwartz-at-trial-java-was-free-android-had-no-licensing-problem/ [https://perma.cc/PT8Q-HFZS] (quoting former Sun CEO testifying that Sun Microsystems welcomed widespread use of the Java programming language and APIs); *see generally* SHAPIRO & VARIAN, *supra* note 54, at 173–93, 196-203.

179. *See* Juan Carlos Perez, *Google Releases Android SDK* [Software Development Kit], MACWORLD (Nov. 12, 2007), www.macworld.com/article/1061005/androidsdk.html [https://perma.cc/JV3Z-4CTX?type=image].

180. *See Congratulations Google, Red Hat and the Java Community!!*, JONATHAN'S BLOG! (Nov. 5, 2007), http://web.archive.org/web/20101023072550/http://blogs.sun.com/jonathan/entry/congratulations_google [https://perma.cc/53KC-GXBJ] (reporting that Schwartz's congratulatory note masked disappointment about Google's unwillingness to enter into a licensing arrangement).

181. *See* Clark & Tuna, *supra* note 23.

182. *See supra* text notes 179–80.

183. The Java Platform, Micro Edition (ME) was launched in late 2006. *See Java Platform, Micro Edition*, WIKIPEDIA, https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition [https://perma.cc/DWR9-E9SM]. One significant difference is that Sun opted to distribute ME using the GNU GPL license.

negotiating table. When licensing negotiations with Google reached an impasse, something had to give.[184]

Oracle's acquisition of Sun brought legal action against Google into play. Larry Ellison, Oracle's co-founder and CEO, had a reputation for brash business tactics.[185] Whereas Sun's leadership had embraced open technology with religious fervor, Oracle's approach had been strategic.[186] Furthermore, Oracle had enjoyed recent success in high stakes copyright enforcement.[187] Oracle's leadership team sought to pursue a far more aggressive Java licensing strategy.

This Part examines the tumultuous history leading up to and through the *Oracle v. Google* litigation as background for understanding the underlying copyright issues. Section III.A explains the technological and industrial context. Section III.B examines the first six years of the *Oracle v. Google* litigation saga. Section III.C discusses the uncertain state of play surrounding API copyright protection in the wake of the *Oracle v. Google* litigation. Part IV critically analyzes the *Oracle v. Google* decisions and explores the policy considerations surrounding copyright treatment of APIs.

A. The Technological and Industrial Context

A confluence of forces set the stage for the *Oracle v. Google* litigation: (1) the development, widespread adoption, and use of the Java programming language for website design; (2) the smartphone revolution and Google's decision to develop an open, distinctive mobile platform using the Java language plus aspects of the Java Standard

184. *See Oracle Buys Sun Microsystems For $7.4B*, CBS NEWS (Apr. 20, 2009), http://www.cbsnews.com/news/oracle-buys-sun-microsystems-for-74b/ [https://perma.cc/ZK3B-PQSV] (reporting that analysts had long said that Sun could not stand on its own and were surprised when merger talks with IBM in late 2008 broke down).

185. *See* Madeline Stone, *Here's How Insanely Competitive Oracle Billionaire Larry Ellison Really Is*, BUS. INSIDER (May 7, 2016), http://www.businessinsider.com/billionaire-larry-ellison-most-competitive-man-2016-5 [https://perma.cc/2F7X-AWZE]; Sarah Lacy, *Larry Ellison Hearsay: "We Can't Be Successful if We Don't Lie to Customers"*, TECHCRUNCH (Dec. 1, 2010), https://techcrunch.com/2010/12/01/larry-ellison-hearsay-we-cant-be-successful-if-we-dont-lie-to-customers/ [https://perma.cc/GJY5-S7EZ].

186. While Oracle opposed strong intellectual property protection for computer software in the early 1990s, it began to build its IP arsenal as IP threats emerged.

187. In 2007, Oracle sued SAP for copyright infringement by one of its subsidiaries. The jury awarded Oracle damages of $1.3 billion in 2010, the largest copyright award in U.S. history. *See* Verne F. Kopytoff, *SAP Ordered to Pay Oracle $1.3 Billion*, N.Y. TIMES (Nov. 23, 2010), http://www.nytimes.com/2010/11/24/business/24oracle.html?mcubz=0 (last visited Jan. 27, 2018); Karen Gullo, *Oracle Wins $1.3 Billion Verdict for Closed SAP Unit's Illegal Downloading*, BLOOMBERG (Nov. 23, 2011), http://www.bloomberg.com/news/articles/2010-11-23/sap-must-pay-oracle-1-3-billion-over-unit-s-downloads (last visited Jan. 27, 2018). Although the trial judge overturned the damages award as excessive, the parties eventually settled for $359 million. *See* Jim Henschen, *Oracle Lawsuit Against SAP Settled at Law*, INFORMATIONWEEK (Nov. 14, 2016), http://www.informationweek.com/cloud/software-as-a-service/oracle-lawsuit-against-sap-settled-at-last/d/d-id/1317483 [https://perma.cc/2GTW-QWQB].

Edition API; and (3) Oracle's acquisition of Sun Microsystems at a critical stage of Android's ascendance. The story illustrates the complex interplay of technological evolution, industry norms, bargaining leverage, ambiguity surrounding the meaning of "open" technology, and lingering uncertainty about the scope of copyright protection for APIs.

1. The Java Story

The Java ecosystem emerged from Sun Microsystems's distinctive — and somewhat quirky — business, technological, and innovative culture.[188]

i. The Corporate Environment: Sun Microsystems in the 1980s and 1990s

In 1982, Stanford University classmates Vinod Khosla, Andy Bechtolsheim, and Scott McNealy and Bill Joy, a University of California at Berkeley computer scientist who played an integral role in developing the Berkeley Software Distribution "(BSD") UNIX operating system,[189] envisioned a breakthrough networked computer engineering workstation.[190] During graduate school and their early careers, they were exposed to the remarkable technologies being developed at the Xerox Palo Alto Research Center: the Alto computer, bitmap displays, and the Ethernet.[191] They formed Sun Microsystems in 1982 to bring their visionary system to the marketplace.

---

188. *See* David Bank, *The Java Saga*, WIRED (Dec. 1, 1995), http://www.wired.com/1995/12/java-saga/ [https://perma.cc/ELE5-ZPG3] (noting that while "Sun's machines had a reputation for being too complicated, too ugly, and too nerdy for mass consumption," its leadership was willing "to loosen[] the reins on some of its most precocious [programmer] talent"); Tekla Perry, *After the Sun (Microsystems) Sets, the Real Stories Come Out*, IEEE SPECTRUM (May 30, 2014), http://spectrum.ieee.org/view-from-the-valley/at-work/tech-careers/after-the-sun-microsystems-sets-the-real-stories-come-out [https://perma.cc/87DY-DUUG].

189. Originally developed by Bell Labs, MIT, and General Electric, UNIX established the foundation for time sharing of mainframe computers. It was historically developed as a closed, proprietary system. The BSD project developed an interoperable version of UNIX, *see Berkeley Software Distribution*, WIKIPEDIA, https://en.wikipedia.org/wiki/Berkeley_Software_Distribution [https://perma.cc/E839-6TPR]; *Bill Joy*, WIKIPEDIA, https://en.wikipedia.org/wiki/Bill_Joy [https://perma.cc/LY7K-BSF3] (featuring a permissive free software licensing framework with minimal restriction of the redistribution of software built on this foundation). *See BSD Licenses*, WIKIPEDIA, https://en.wikipedia.org/wiki/BSD_licenses [https://perma.cc/A2NR-VTV3]. The BSD license diverged from the viral, open source (sometimes referred to as "copyleft") licenses that require that software built on open source code be made available to other developers on an open source basis — the so-called share-alike requirement. *See* Carver, *supra* note 63.

190. *See* Perry, *supra* note 188; *William Joy (1954–), Programmer; Founder of Sun Microsystems*, *in* THE INTERNET: BIOGRAPHIES 138 (Hilary W. Poole ed., 2005).

191. *See PARC (company)*, WIKIPEDIA, https://en.wikipedia.org/wiki/PARC_(company) [https://perma.cc/B5Z3-FB7Q]; *Sun Microsystems*, WIKIPEDIA, https://en.wikipedia.org/

Sun hit profitability in its first quarter of operations and quickly developed a reputation for high performance, networked UNIX-based workstations with high-quality graphics.[192] Their technology fueled Silicon Valley's meteoric rise. Although less widely known than Apple, Microsoft, or IBM because its products were sold to other technology companies rather than the general public, Sun nevertheless commanded the respect of the high technology sector. Sun expanded into processors and servers and became one of the world's most successful technology companies. Sun went public in 1986 under the stock symbol SUNW, for Sun Workstations (later Sun World-Wide),[193] and hit $1 billion in revenues in 1988, a record for a Silicon Valley company.[194] Thanks to its reputation for cutting-edge products and an engineer-friendly culture, the company attracted a talented, eclectic, and loyal group of engineers and programmers.

Sun's revenues and market value grew steadily from its founding into the mid-1990s and skyrocketed during the dot-com boom.[195] Flush with venture capital investment, many start-ups wanted the best workstations and servers for their engineering and programming teams. Sun's outlook was bright as the Internet Age commenced.

ii. Development of Java

Sun's foray into developing a new programming language began in 1990 as a skunkworks project.[196] Triggered by an effort to retain a top programmer, the initiative aimed initially at developing a new generation of software to replace Sun's C++ and C APIs and tools.[197] Sun's leaders recognized that the success of the project required that the elite team be insulated from the rest of Sun's operations, especial-

---

wiki/Sun_Microsystems [https://perma.cc/CY4T-32DG]; MICHAEL A. HILTZIK, DEALERS OF LIGHTNING: XEROX PARC AND THE DAWN OF THE COMPUTER AGE (2000).

192. *See Sun Microsystems*, WIKIPEDIA, https://en.wikipedia.org/wiki/Sun_Microsystems [https://perma.cc/CY4T-32DG].

193. *See id.*

194. *See William Joy (1954–), Programmer*, *supra* note 190.

195. *See Sun Microsystems*, WIKIPEDIA, https://en.wikipedia.org/wiki/Sun_Microsystems [https://perma.cc/CY4T-32DG]; Lee Devlin, *The Sun Also Sets*, K0LEE.COM (Oct. 2, 2009), http://k0lee.com/2009/10/sun-also-sets/ [https://perma.cc/UGW7-Z5F8] (tracing Sun's meteoric stock rise from 1982 to 2000, and fall).

196. A skunkworks project refers to "a project developed by a small and loosely structured group of people who research and develop a project primarily for the sake of radical innovation." *See Skunkworks Project*, WIKIPEDIA, https://en.wikipedia.org/wiki/Skunkworks_
project [https://perma.cc/A8AH-Y65R]. The term, derived from the name of the moonshine factory in the Li'l Abner comic book series, traces to Lockheed's World War II Advanced Developments Program.

197. *See* Bank, *supra* note 188; *History of the Java™ Programming Language*, WIKIBOOKS, https://en.wikibooks.org/wiki/Java_Programming/History [https://perma.cc/36RD-33ES].

ly the business pressures to meet quarterly targets.[198] This so-called "Green Project" team took up residence in rented office space elsewhere in Silicon Valley.[199]

The project evolved into developing a computer language and handheld device that could be used for both digitally controlled consumer products (such as televisions) and computers.[200] Such a language needed to be scaled for embedded systems — computer systems with a dedicated function within other systems.[201] The team initially focused on developing a distributed computing environment for set-top boxes, interactive TVs, and video cassette recorders through a wireless network.[202] Such a system would have more limited functionality than general purpose computers and requires a more compact footprint.

James Gosling took the lead in developing the software.[203] He designed a secure, reliable, object-oriented,[204] platform-independent language that could interpret other languages and function on small computer chips embedded in consumer devices. By 1993, the software (code-named Oak) was integrated into a versatile device that could work with interactive TV technology, but Sun was unable to interest consumer electronics or cable companies.[205]

Just when the project looked doomed, Bill Joy saw the opportunity to adapt Gosling's software for the nascent, but promising, World Wide Web.[206] Joy realized that Oak could be re-purposed to program webpages, as opposed to consumer devices. The team convinced Sun to pump more resources into the project.[207] "Java," the renamed project, aimed to develop a simple, lean, platform-independent, real-time, embeddable, multi-tasking programming language for web functionality. Java had a similar syntax to the widely-used C language, but was far more compact, efficient, and secure. Of perhaps greatest importance, Java enabled "write once, run anywhere" ("WORA") functionality: Java applets could run on Apple, Windows, or UNIX machines without any customization. Java also enabled real-time interactivity, multimedia, and animation, which greatly enhanced the

198. *See* Bank, *supra* note 188.

199. *Id.*

200. *See History of the Java Programming™ Language*, *supra* note 197.

201. *See Embedded System*, WIKIPEDIA, https://en.wikipedia.org/wiki/Embedded_system [https://perma.cc/CDQ5-7K2S].

202. *See James Gosling (1956-), Inventor of Java*, *in* THE INTERNET: BIOGRAPHIES 132–36 (Hilary W. Poole ed.)., 2005).

203. *See id.*

204. *See Object-oriented Programming*, WIKIPEDIA, https://en.wikipedia.org/wiki/Object-oriented_programming [https://perma.cc/FCW5-HVB5].

205. *See* Bank, *supra* note 188; *James Gosling, supra* note 202.

206. *See William Joy (1954–), Programmer*, *supra* note 190.

207. *See* JOHN HUNT, JAVA FOR PRACTITIONERS: AN INTRODUCTION AND REFERENCE TO JAVA AND OBJECT ORIENTATION 49 (2012).

dynamism of webpages. Java added new dimensions to Web functionality. Java applets enabled users to interact with websites in new and exciting ways.

Gosling built Java as an object-oriented programming ("OOP") language and platform, utilizing a powerful programming paradigm that was gaining salience in the programming community in the early 1990s.[208] In contrast to conventional procedural programming languages such as C, Fortran, Pascal, and Basic, which break tasks down into a structured series of computational steps,[209] OOP models tasks using relational objects that expose behavior (methods) and data (members or attributes) using interfaces.[210] The OOP paradigm offered various programming efficiencies, such as reusability and ease of modification and maintenance.[211]

With the experimental new software platform reaching fruition, Sun faced a difficult business strategy choice. Although Sun had always been a proponent of open standards for software interfaces,[212] this project would require the free release of a software implementation — that is, the full program. Marc Andreessen,[213] the University of Illinois wunderkind who created the pioneering Mosaic web browser,[214] had released Mosaic for free for noncommercial use, but major companies were not yet in the business of giving away source code. Many in the industry coveted source code as the crown jewels of high technology businesses and were loath to share it.[215]

Eric Schmidt, Sun's Chief Technology Officer who had assured the "Green" team that they would be insulated from the business managers, was at the center of an impending corporate storm. As he would later describe:

---

208. *See Object-oriented Programming*, *supra* note 204.

209. *See Procedural Programming*, WIKIPEDIA, https://en.wikipedia.org/wiki/Procedural_programming [https://perma.cc/3ZBB-649B].

210. *See Object-oriented Programming*, WIKIPEDIA, https://en.wikipedia.org/wiki/Object-oriented_programming [https://perma.cc/FCW5-HVB5].

211. *See Advantages and Disadvantages of Object-Oriented Programming (OOP)*, THE SAYLOR FOUNDATION http://www.saylor.org/site/wp-content/uploads/2013/02/CS101-2.1.2-AdvantagesDisadvantagesOfOOP-FINAL.pdf [https://perma.cc/MNL9-NDSJ].

212. Sun Microsystems has been the leading member of the American Committee for Interoperable Systems ("ACIS"), an early lobbying organization advocating open platforms. *See* BAND & KATOH, *supra* note 10, at 308 (noting that Peter Choy, who headed ACIS, worked for Sun).

213. *See Marc Andreessen*, WIKIPEDIA, https://en.wikipedia.org/wiki/Marc_Andreessen [https://perma.cc/Q2VU-X9E9].

214. *See Mosaic (web browser)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Mosaic_(web_browser) [https://perma.cc/E4Q4-BFMJ].

215. *See* Eugene A. Feher & Dmitriy S. Andreyev, *Source Code in Patent Litigation*, LAW360 (Apr. 30, 2008) http://www.law360.com/articles/54750/source-code-discovery-in-patent-litigation [https://perma.cc/3EMY-GPWG] (noting that "most companies consider their source code to be highly confidential and part of the 'crown jewels' of the company" and that "[s]ource code frequently contains secret proprietary algorithms that provide a vital competitive advantage").

> The conversation that never took place, but that I could feel all around me, was, 'Eric, you are violating every principle in the company. You are taking our technology and giving it away to Microsoft and every one of our competitors. How are you going to make money?' At the time, I didn't have an answer. I would make something up. I would lie. What I really believed was that Java could create an architectural franchise. The quickest way was through volume and the quickest way to volume was through the Internet.[216]

Sun secretly invited a select group of programmers to test Java in December 1994.[217] The test revealed that the WORA functionality was a game-changer and word of Java's capabilities spread like wildfire throughout the programmer community.[218]

Sun officially launched Java in January 1995. The business strategy epiphany came when Marc Andreessen, the new CEO of Netscape and developer of Netscape's breakthrough Navigator browser,[219] raved to the SAN JOSE MERCURY NEWS: "What these guys are doing is undeniably, absolutely new. It's great stuff. There's so much stuff people want to do over the network that they haven't had the software to do. These guys are really pushing the envelope."[220]

Having already released Java to a select programmer audience, Sun decided to focus on establishing Java as the standard language for web development and figure out how to make money later. It followed the "'profitless' approach to building market share" that Netscape had employed in giving away its Navigator browser.[221] As Joy would later remark, "There was a point at which I said, 'Just screw it, let's give it away.' Let's create a franchise."[222]

Due in part to the robust performance of its hardware divisions,[223] Sun could afford to take more risk with the revenue side of its soft-

---

216. *See* Bank, *supra* note 188.

217. *See* Bank, *supra* note 188; *William Joy (1954–), Programmer*, *supra* note 190.

218. *See* Bank, *supra* note 188 (reporting that release of early versions of Java in December 1994 "unleashed stratospheric expectations"); *William Joy (1954–), Programmer*, *supra* note 190.

219. *See Netscape Navigator*, WIKIPEDIA, https://en.wikipedia.org/wiki/Netscape_Navigator [https://perma.cc/4DT2-UB3E].

220. *See* David Bank, *Why Sun Thinks Hot Java Will Give You a Lift New Software Designed to Make World Wide Web's 'Home Pages' More Useful; And Spur Computer Sales*, SAN JOSE MERCURY NEWS 1A (Mar. 23, 1995); Bank, *supra* note 188 (quoting Kim Polese, Java's senior product manager: "That quote was a blessing from the god of the Internet").

221. *See* Bank, *supra* note 188.

222. *See id.*

223. *See id.* (reporting that Sun's annual revenues from its hardware products were expected to exceed $6 billion in 1995).

ware business. Its larger concern, as manifested in the years ahead, was in preventing Microsoft from dominating the emerging Internet marketplace in the same way it had dominated desktop computing software.[224] Scott McNealy, Sun's fiercely competitive CEO, imagined that "disposable word processors and spreadsheets delivered over the Web via Java, priced per use" could "blow[] up Gates's lock [on the desktop software marketplace] and destroy[] his mode of shrink-wrapped software that runs only on his platform."[225] The WORA approach promised to invigorate the software competition landscape.[226]

In May 1995, Netscape licensed Java as part of its market-leading Navigator browser.[227] Although Sun authorized Netscape's use for a pittance,[228] it foresaw that this move would produce rapid diffusion across the programming community and the Web. Sun also provided Java for free to noncommercial users.[229] Java's ability to transform static webpages into engaging, animated, interactive websites revolutionized web design within a matter of months.[230]

Sun was especially concerned that Microsoft would leverage its eighty percent share of the desktop software marketplace to control Internet software development.[231] In March 1995, Microsoft announced "Blackbird," a new Web development package slated for a January 1996 release, that would contain an application programming language configured to work with Microsoft software.[232] In response, Sun actively pursued below-cost licensing deals in an effort to prevent Microsoft from burying the competition.[233] At the same time, Microsoft was pressuring other companies to withdraw support for Java.[234]

As Blackbird languished (and ultimately never launched),[235] Microsoft shifted its Internet strategy. By late 1995, Sun and Microsoft

---

224. *See id.* (noting Sun co-founder and CEO Scott McNealy's rivalry with Bill Gates).

225. *See id.* (first quotation Bank's paraphrase of McNeely; second quotation from McNealy).

226. *See* Mark A. Lemley & David McGowan, *Could Java Change Everything? The Competitive Propriety of a Proprietary Standard*, 43 ANTITRUST BULL. 715 (1998).

227. *See William Joy (1954–), Programmer*, *supra* note 190.

228. *See* Bank, *supra* note 188 (reporting that Netscape "paid a paltry US$750,000" to license without any per-copy charges).

229. *See id.*

230. *See William Joy (1954–), Programmer*, *supra* note 190.

231. *See* Bank, *supra* note 188 (quoting Michael Sheridan, an original member of the "Green Project" team and Java business strategist, that "Sun's window is six to twelve months. [We] need to move quickly because Microsoft will respond in a way that freezes development.").

232. *See Blackbird (online platform)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Blackbird_(online_platform) [https://perma.cc/GT37-R2PR].

233. *See* Bank, *supra* note 188 (quoting Eric Schmidt: "This loses money in the licensing business for the foreseeable future. It's a strategic investment in market share.").

234. *See* United States v. Microsoft Corp., 84 F. Supp. 2d 9 (D.D.C 1999) (declaring findings of fact).

235. *See Blackbird (online platform)*, *supra* note 232.

worked out the basis for a license agreement.[236] In March 1996, Sun agreed to a Technology License and Distribution Agreement ("TLDA") that allowed Microsoft to use, modify and adapt Java technology in developing MS Internet Explorer 4.0 and other software products.[237] In keeping with its WORA interoperability principle, the TLDA required Microsoft to adhere to Java's standardized application environment and compliance tests.[238]

To live up to Java's initial high praise and build momentum, Sun expanded its Java development efforts. It rolled out the first stable Java Development Kit in early 1996 and continued to expand features over the following year.[239] The Java language comprises words, symbols, and pre-written programs to carry out various commands, such as printing something on the screen or performing a basic mathematical calculation. Sun organized sets of pre-written programs (methods, which are grouped in classes) into API packages (or class libraries). Each API package reflects a set of declarations[240] or functional specifications needed to invoke the methods. It is executed through detailed implementing code. Although a Java programmer can also write new code (methods) from scratch, the pre-written methods within the Java API packages provide convenient, efficient, reliable, standardized building blocks, thereby saving Java programmers tremendous tedious effort.

Sun's strategy succeeded in establishing Java as a de facto industry standard. By the end of 1996, Apple, IBM, Netscape, Oracle, and more than a hundred other companies had committed to the Java platform through the "100% Pure Java" initiative.[241] By that time, Sun employed three hundred people in its JavaSoft division and approximately thirty-five percent of websites used Java. The applets could be viewed on UNIX, Windows, Apple, or DOS computers.

Sun's respect for its programmer culture, and its effort to harness network effects and thereby outmaneuver Microsoft, pushed Java onto an open development path. Sun's highly profitable hardware division afforded its Java division flexibility to operate as a loss leader. As one industry observer presciently noted in late 1995, "Java is unlikely ever

---

236. *See* Sun Microsystems, Inc. v. Microsoft Corp., 21 F. Supp. 2d 1109, 113 (N.D. Cal. 1998).

237. *See id.* at 113–14.

238. *See id.* at 114; *see also Technology Compatibility Kit*, WIKIPEDIA, https://en.wikipedia.org/wiki/Technology_Compatibility_Kit [https://perma.cc/V3KP-9BQS] (describing the Java Compatibility Kit (JCK) used to ensure that implementations are compatible with the Java platform).

239. *See Java version history*, WIKIPEDIA, https://en.wikipedia.org/wiki/Java_version_history [https://perma.cc/Q6FC-889D].

240. *See* The Java™ Tutorials — Declaring Classes, ORACLE, https://docs.oracle.com/javase/tutorial/java/javaOO/classdecl.html [https://perma.cc/3DXU-KNTJ].

241. *See* Paul Floren, *Sun's Java: Can It Burn Microsoft?*, INT'L HERALD TRIBUNE, Jan. 20, 1997.

to become a major profit center at Sun, though any increase in Web traffic is bound to increase sales of Sun's workstations and servers."[242]

As part of its effort to establish Java as the standard programming language for the Internet, Sun proposed to the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) in March 1997 that the Java "platform" — consisting of the Java language, class file format, byte codes recognized by the Java Virtual Machine, and Java APIs — be formally designated a *de jure* international standard.[243] The process bogged down as a result of concerns among members of the Joint Technical Committee regarding the appropriateness of a single firm seeking standard approval for their product and whether such a firm should be permitted to retain intellectual property rights in the proposed standard.[244]

Microsoft's deployment of its own version of Java, compatible only with other Microsoft products in violation of the WORA principle, threatened Sun's Java development strategy. After Microsoft distributed its Internet Explorer 4.0 browser program without components of the Java System Developer Kit 1.1 in October 1997, Sun sued Microsoft for breach of contract, trademark infringement, copyright infringement, false advertising, and unfair competition.[245] These allegations coincided with and reinforced antitrust concerns about Microsoft's business practices.[246]

Of principal importance for the API copyright issue, the Microsoft threat pushed Sun to pursue an aggressively open Java development strategy that encouraged widespread adoption as well as adherence to the WORA principle.[247] Sun ultimately withdrew from efforts to seek formal standardization of Java out of concern that it would have to cede too much control over Java's development path to other entities, including competitors who might not share Sun's vi-

---

242. *See* Bank, *supra* note 188.

243. *See* Tineke M. Egyedi, *Why Java™ Was - Not - Standardized Twice*, IEEE PROCEEDINGS OF THE 34TH HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES (2001).

244. *See* Lemley & McGowan, *supra* note 226, at 755.

245. *See* John Markoff, *Sun Sues Microsoft in Dispute Over Java*, N.Y. TIMES (Oct. 8, 1997), http://politics.nytimes.com/library/cyber/week/100897java.html [https://perma.cc/8WBG-HJ6F].

246. *See* United States v. Microsoft Corp., 87 F. Supp. 2d 30, 44 (D.D.C. 2000), *aff'd in part, rev'd in part per curiam*, 253 F.3d 34 (D.C. Cir. 2001) (en banc); John E. Lopatka & William H. Page, *Antitrust on Internet Time: Microsoft and the Law and Economics of Exclusion*, 7 SUP. CT. ECON. REV. 157 (1999); William H. Page & John E. Lopatka, *The Dubious Search for "Integration" in the Microsoft Trial*, 31 CONN. L. REV. 1251 (1999).

247. *See* Peter Wayner, *What the Battle Over Java Is Really About*, N.Y. TIMES (Oct. 11, 1997), http://politics.nytimes.com/library/cyber/week/101197java.html [https://perma.cc/65LS-F87G].

sion.[248] Nonetheless, the Microsoft threat committed Sun to an open development path for Java.

In 1998, Sun released the Java 2 Standard Edition Platform. It contained eight API packages, three of which — java.lang, java.io, and java.util — were necessary to use the Java programming language.[249] In the following years, Sun gradually expanded the number of API packages, classes, and methods.

Sun also established the Java Community Process ("JCP") in 1998 to enable users to participate in the development of standard technical specifications for Java technology.[250] Community members were invited to propose Java Specification Requests ("JSRs") for expanding and updating the Java platform. The JCP reviews JSRs through a public process akin to administrative rulemaking. The JCP Executive Committee,[251] comprised of major stakeholders, decides whether to approve JSRs.

One of the goals of the JCP was to bring order to the emerging, but fragmented, mobile device ecosystem. The mobile marketplace was taking off in the mid-1990s with a variety of personal digital assistants ("PDAs"),[252] cell phones, and other consumer devices. In 1998 and 1999, Sun coalesced the various interests through the JCP in developing the Java 2 Micro Edition ("J2ME").[253] Many cell phone developers licensed the J2ME Platform for their products.

After four years of tumultuous litigation,[254] Sun and Microsoft settled their litigation in January 2001.[255] Microsoft agreed to pay Sun $20 million and was permanently prohibited from using "Java compatible" trademarks on its products.[256] The copyright infringement allegations relating to APIs were not pursued.

---

248. *See* Lemley & McGowan, *supra* note 226, at 770.

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

250. *See Java Community Process*, WIKIPEDIA, https://en.wikipedia.org/wiki/Java_Community_Process [https://perma.cc/NN53-WMJY].

251. *See JCP Executive Committee*, WIKIPEDIA, https://en.wikipedia.org/wiki/JCP_Executive_Committee [https://perma.cc/7GN3-4SSN].

252. Palm successfully introduced the Palm Pilot in 1997, but gradually lost market share as new devices, such as Research in Motion's BlackBerry, became popular. *See Palm, Inc.*, WIKIPEDIA, https://en.wikipedia.org/wiki/Palm,_Inc. [https://perma.cc/78Y8-MJD8]; *BlackBerry*, WIKIPEDIA, https://en.wikipedia.org/wiki/BlackBerry [https://perma.cc/7BED-8JV7].

253. *See J2ME Programming/The J2ME Platform*, WIKIBOOKS, https://en.wikibooks.org/wiki/J2ME_Programming/The_J2ME_Platform. [https://perma.cc/YQF4-J6AK].

254. *See* Sun Microsystems, Inc. v. Microsoft Corp., 21 F. Supp. 2d 1109 (N.D. Cal. 1998) (granting preliminary injunction enjoining Microsoft from distributing any software implementing Java), *vacated*, 188 F.3d 115 (9th Cir. 1999), *reinstating injunction*, 87 F. Supp. 992 (N.D. Cal. 2000).

255. *See* Stephen Shankland, *Sun, Microsoft Settle Java Suit*, CNET (Mar. 15, 2002), http://www.cnet.com/news/sun-microsoft-settle-java-suit/. [https://perma.cc/R2DV-Z4NK].

256. Sun would later prevail in a separate antitrust and patent infringement action against Microsoft resulting in an award of $1.6 billion. *See* Pruitt & Roberts, *supra* note 18; Stephen Shankland, *Sun brings antitrust suit against Microsoft: The company files a private antitrust*

iii. The Setting Sun

Sun's sales collapsed following the dot-com bubble burst in early 2000. Many of the dot-com companies that had ordered Sun hardware went bankrupt, causing new orders to plummet and driving workstation and server prices downward as failed start-ups auctioned off their assets to repay creditors. Sun's stock went into freefall.

As the Silicon Valley economy recovered in 2004, advanced microcomputers displaced demand for far more costly Sun workstations. Sun cancelled major processor projects, closed one of its two major factories, and initiated a series of layoffs. Sun's hardware business somewhat stabilized after 2005, but prospects for future growth were bleak. To expand Java's reach, Sun licensed Java, including its Standard Edition, Enterprise Edition, and Micro Edition, under the GNU GPLv2 in 2006.[257]

Symbolizing its shift in direction, Sun changed its Nasdaq Stock Market ticker in August 2007 from SUNW to JAVA.[258] As the press release highlighted, "[t]he new ticker reflects Sun's 12-year-old Java programming language, which is available free . . . There are 6 million Java developers, and the language is used in 5.5 billion devices, including personal computers and mobile phones."[259] In his accompanying blog post, Jonathan Schwartz proudly proclaimed that:

> Java touches nearly everyone — *everyone* — who touches the internet. Hundreds of millions of users see Java, and its ubiquitous logo, every day. On PC's, mobile phones, game consoles — you name it, wherever the network travels, the odds are good Java's powering a portion of the experience . . .
>
> I know that sounds audacious, but wherever I travel in the world, I'm reminded of just how broad the opportunity has become, and how pervasively the technology and brand have been deployed. Java truly is everywhere.

---

suit against Microsoft seeking damages that could top $1 billion, CNET (Jul. 20, 2002), https://www.cnet.com/news/sun-brings-antitrust-suit-against-microsoft-1/ [https://perma.cc/U6PD-DS3L].

257. *See Sun to Open-Source Java Under GPL*, PRACTICAL TECH. (Nov. 11, 2006), http://practical-tech.com/development/sun-to-open-source-java-under-gpl/415/ [https://perma.cc/XEC2-YC5B]. The GNU GPL requires that software built on the open source code base be available to others on an open source basis — the so-called share-alike requirement. *See* Carver, *supra* note 63.

258. *See Sun Microsystems' New Ticker: JAVA*, L.A. TIMES (Aug. 24, 2007), http://articles.latimes.com/2007/aug/24/business/fi-wrap24.s4 [https://perma.cc/3D7D-NK2D].

259. *See id.*

> Ask a teenager if they know Java, and they'll point to their favorite mobile applications, the video uploader for their social network, or their game console. As for working professionals, I had dinner with a financial analyst a few months ago who said he saw the Java launch experience "a few times a day" when accessing intranet applications — as did tens of thousands of his fellow employees. Daily. Global companies like Google and eBay (and Vodafone and Citigroup) are built on Java, every major PC manufacturer bundles Java upon shipment, as does every mobile phone manufacturer, and tens of millions of developers touch it every day in the world's IT shops. Students learn it to get college credits for computer science, and there are more Java courses on university campuses than we ever imagined. Wherever it goes, Java brings limitless opportunity — to Sun, and to our partners that develop, use or deploy it.
>
> . . . SUNW represents the past, and [it's] not without a nostalgic nod that we've decided to look ahead.
>
> JAVA is a technology whose value is near infinite to the internet, and a brand that's inseparably a part of Sun (and our profitability) . . . .[260]

Sun initially succeeded in gaining wide adoption of the Java Micro Edition platform for feature phones — mobile phones with limited capability, principally voice and text messaging with basic multimedia and rudimentary internet access.[261] It failed, however, to develop a robust revenue stream and suffered further deep losses during the 2008 financial crisis. Sun's market value fell eighty percent between November 2007 and November 2008, resulting in further substantial layoffs.[262] By this point, Sun's leadership viewed its software busi-

---

260. *See* Jonathan I. Schwartz, *The Rise of JAVA — The Retirement of SUNW*, JONATHAN SCHWARTZ BLOG (Aug. 23, 2007), https://jonathanischwartz.wordpress.com/2007/08/23/the-rise-of-java-the-retirement-of-sunw/ [https://perma.cc/3TST-RH6E] (emphasis in original).

261. *See Feature Phone*, WIKIPEDIA, https://en.wikipedia.org/wiki/Feature_phone [https://perma.cc/9T8F-9NDA].

262. *See* Ashlee Vance, *Sun Microsystems Reports $1.7 Billion Loss and Falling Sales*, N.Y. TIMES (Oct. 30, 2008), at B3, http://www.nytimes.com/2008/10/31/technology/companies/31sun.html (last visited Jan. 27, 2018); Lee Devlin, *The Sun Also Sets*,

nesses, revolving around Java, as the company's future. They came to see developing a robust licensing model as essential to the company's prosperity, and possibly its survival.

2. Google, the Mobile Computing Revolution, and Development of Android

Just as Sun was reaching its highest point during the dot-com bubble, Sergey Brin and Larry Page were developing a search engine that would become the next shining star.[263] Drawing on the Navigator and Java strategies, Google focused on widespread adoption rather than revenue generation. It offered free access to its simple, no-nonsense search engine. As the technology press recognized its "uncanny knack for returning extremely relevant results,"[264] Google amassed loyal users and separated itself from the crowded field of search engines. Unlike Netscape and Sun, however, Google developed a robust revenue model for its "free"-to-users software: keyword advertising. By October 2000, just as Sun's hardware business was setting, Google launched its AdWords program.[265] In August 2001, Google named Eric Schmidt, Sun's former CTO, as its CEO. The press touted that Schmidt had "led the development of Java, Sun's platform-independent programming technology, and defined Sun's Internet software strategy."[266]

With revenue flowing from AdWords, Google developed a series of new search projects — images, news, shopping, Gmail, maps — which reinforced and expanded its advertising business. Google went public in 2004[267] and continued to expand its reach with Google Books, YouTube, and other projects.[268]

K0LEE.com (Oct. 2, 2009), http://k0lee.com/2009/10/sun-also-sets./ [https://perma.cc/UGW7-Z5F8].

263. Ironically, Andy Bechtolsheim, one of Sun's co-founders, was among the first to recognize Google's promise. In August 1998, he wrote the founders a check for $100,000 before the company was established. *See* Tony Long, *Sept. 7, 1998: If the Check Says 'Google Inc., 'We're 'Google Inc.,'* WIRED (Sept. 7, 2007), http://www.wired.com/2007/09/dayintech-0907/ [https://perma.cc/5HS8-N9N3]. It would prove to be one of the wisest investments in Silicon Valley history. *See Andy Bechtolsheim*, WIKIPEDIA, https://en.wikipedia.org/wiki/Andy_Bechtolsheim [https://perma.cc/VF6R-MMQ9] (estimating that Bechtolsheim's $100,000 investment in 1998 was worth approximately $1.7 billion by March 2010). Google's stock has more than doubled again since 2010.

264. *See Top 100 Web Sites: Search Engines*, PC MAGAZINE, Feb. 9, 1999, at 118.

265. *See AdWords*, GOOGLE, https://en.wikipedia.org/wiki/AdWords [https://perma.cc/X4BR-R4SX].

266. *See Google Names Dr. Eric Schmidt Chief Executive Officer*, NEWS FROM GOOGLE (Aug. 6, 2001), http://googlepress.blogspot.com/2001/08/google-names-dr-eric-schmidt-chief.html [https://perma.cc/5365-UNWT].

267. *See* John Markoff, *THE GOOGLE I.P.O.: THE OVERVIEW; Google's Sale of Its Shares Will Defy Wall St. Tradition*, N.Y. TIMES (Apr. 30, 2004),

Google's leaders foresaw the next gathering wave: smartphones and mobile platforms.[269] The mobile marketplace, however, was a morass of telecommunication companies, handset makers, and software providers.[270] The telecommunications companies (telcos) were notoriously protective of their networks.[271] The handset makers, commonly referred to as original equipment manufacturers ("OEMs"), had divergent strategies and business models. The widespread feature phones had little capability to access the Internet. RIM's BlackBerry phone, geared for business customers, had proven the robust demand for mobile Email devices, but did not offer fully functioning web browsing capability.[272] Microsoft and Symbian were promoting proprietary mobile operating systems but without notable success. Google executives worried, however, that Microsoft could gain traction and ultimately steer consumers away from Google search and other services.[273]

Just as the Internet's open architecture had brought order and innovation, Google's leaders came to see that an open source platform for mobile communications could provide a comparably important platform for the growing shift to portable, hand-held devices.[274] They began to recognize that leading this transformation could pay large

---

http://www.nytimes.com/2004/04/30/business/google-ipo-overview-google-s-sale-its-shares-will-defy-wall-st-tradition.html (last visited Jan. 27, 2018).

268. *See Our History in Depth*, GOOGLE, https://www.google.com/about/company/history [https://perma.cc/A9XC-WZR7].

269. In its 2005 10-K filing, Google identified the emerging mobile marketplace as a potential threat to its profitability (emphasis in original):

> ***More individuals are using non-PC devices to access the Internet, and versions of our web search technology developed for these devices may not be widely adopted by users of these devices***.
>
> The number of people who access the Internet through devices other than personal computers, including mobile telephones, handheld calendaring and email assistants, and television set-top devices, has increased dramatically in the past few years. The lower resolution, functionality and memory associated with alternative devices make the use of our products and services through such devices difficult. If we are unable to attract and retain a substantial number of alternative device users to our web search services or if we are slow to develop products and technologies that are more compatible with non-PC communications devices, we will fail to capture a significant share of an increasingly important portion of the market for online services.

Google Inc., Commission Annual Report (Form 10-K) (Mar. 16, 2006) at 32.

270. *See* FRED VOGELSTEIN, DOGFIGHT: HOW APPLE AND GOOGLE WENT TO WAR AND STARTED A REVOLUTION 48–50 (2013).

271. *See* John Markoff, *I, Robot: The Man Behind the Google Phone*, N.Y. TIMES (Nov. 4, 2007), http://www.nytimes.com/2007/11/04/technology/04google.html (last visited Jan. 27, 2018).

272. *See* VOGELSTEIN, *supra* note 270, at 53.

273. *See id.* at 51.

274. *See id.* at 49–53.

dividends for Google's search and other information services. Such an initiative, however, posed serious challenges.

In 2003, Larry Page and Sergey Brin were smitten with the T-Mobile Sidekick, a nifty mobile device designed by Andy Rubin, a former Apple engineer.[275] Page and Brin were especially impressed by the way in which Sidekick provided an authentic web browsing experience.[276] Other mobile devices, such as the BlackBerry, only showed text. Therefore users could not click on Google search ads.[277] Page admired Sidekick's engineering and was pleased that Rubin had adopted Google as the default search engine.[278]

Rubin co-founded Android in October 2003 to develop "smarter mobile devices that are more aware of [their owners'] location and preferences."[279] When Rubin reached out to Page in 2005 to set up a meeting, Page was eager to hear what Rubin had to say. Rubin explained that phones with computer capabilities were the future and that Android was working toward an open platform.[280] This pitch coincided with Google's corporate philosophy and aspirations. In July 2005, Google acquired Android for $50 million, brought Rubin's team on board, and put Rubin in charge of its new mobile division.[281]

Building an open mobile communications platform posed substantial challenges.[282] A new operating system would need to be optimized for the small chips on which handsets were based. The devices would have to work in real time. The platform had to be compact and optimized to the particular functionalities consumers would demand.

In addition, the licensing model had to balance openness with downstream competition and innovation. Google did not believe that the GNU GPL would provide sufficient flexibility for the range of

---

275. *See* John Markoff, *Where Does Google Plan to Spend $4 Billion?*, N.Y. TIMES (Aug. 22, 2005), http://www.nytimes.com/2005/08/22/technology/where-does-google-plan-tospend-4-billion.html (last visited Jan. 27, 2018) (observing that Page and Brin wore the Sidekick all-purpose voice and data communicators on their belts several years ago and that Page had long envisioned a Google-branded smartphone).

276. *See* VOGELSTEIN, *supra* note 270, at 52–53.

277. *See id.* at 53.

278. *See id.* at 53.

279. *See* Ben Elgin, *Google Buys Android for Its Mobile Arsenal*, BUS. WK (Aug. 17, 2005), http://tech-insider.org/mobile/research/2005/0817.html [https://perma.cc/PAZ7-WVP9].

280. *See* VOGELSTEIN, *supra* note 270, at 49 (explaining that:

> [T]he software industry for mobile phones was one of the most dysfunctional in all technology. There wasn't enough bandwidth for users to surf the Internet on a phone without frustration. Phones weren't powerful enough to run anything by rudimentary software. But the biggest problem . . . was that the industry was ruled by an oligopoly.

).

281. *See* John Markoff, *Where Does Google Plan to Spend $4 Billion?*, N.Y. TIMES (Aug. 22, 2005), http://www.nytimes.com/2005/08/22/technology/where-does-google-plan-tospend-4-billion.html (last visited Jan. 21, 2018).

282. *See* VOGELSTEIN, *supra* note 270, at 53.

players it believed would be needed to establish a robust new mobile platform. Google worried that the viral share and share alike provision would discourage handset makers and telcos from making investments in innovative features. A more permissive licensing model, in which downstream suppliers could make proprietary extensions on top of the base platform, would better promote robust competition and innovation.[283]

Google and its newly hired Android team also believed that they would need to create an application programming environment that was familiar and easy to use.[284] At the first high-level Android planning meeting, convened on July 26, 2005, the newly established Android team and Google leaders focused on three questions:

> • Which type of Open Source are we?
> • How do we interact with the OSS [open source software community]?
> • How do we Open Source our JVM [Java Virtual Machine]?[285]

The group envisioned Android "as the world's first Open Source handset solution with built-in Google applications."[286] Google would work closely with telcos and OEMs. Telcos would benefit from "the ability to quickly deploy differentiating features and applications."[287] OEMs would benefit from a "robust, free consumer [open source] platform."[288] And Google "benefits by having control of the user experience and built-in Google apps."[289] Open source was seen as a crit-

283. *See* Email from Andy Rubin to Bob Lee (Aug. 11, 2007), Trial Ex. 230, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012) (No. C 10-03561 WHA) (noting that "[t]he problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it."). In a complex and controversial twist, Google's use of Linux kernel in Android, which is licensed under the GNU GPL, arguably does not trigger the share and share alike licensing requirement. *See* HEATHER J. MEEKER, OPEN (SOURCE) FOR BUSINESS: A PRACTICAL GUIDE TO OPEN SOURCE SOFTWARE LICENSING ch. 8 (2015) (discussing the GPL 2 Border Dispute).

284. Even beyond these challenging issues, smartphone technology was a patent minefield. *See Smartphone Patent Wars*, WIKIPEDIA, https://en.wikipedia.org/wiki/Smartphone_patent_wars [https://perma.cc/8FBQ-ZDT5]. In the previous decades, telcos, OEMs, and software companies had patented a wide range of mobile communication-related technologies. Google would spend billions of dollars acquiring mobile technology patents and defending patent lawsuits. Those issues, however, were not prominent on Google's radar screen as it embarked on its mobile technology odyssey, but they would loom large in the years ahead. *See* VOGELSTEIN, *supra* note 270, at 53.

285. *See* Android GPS [Google Product Strategy]: Key strategic decisions around Open Source at 2 (July 26, 2005), Trial Ex. 1, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012) (No. C 10-03561 WHA).

286. *See id.* at 4.

287. *See id.* at 5.

288. *See id.*

289. *See id.*

ical feature for three reasons: it was capable of (1) disrupting the closed and proprietary nature of the Microsoft and Symbian platforms, leading candidates for a smartphone platform at the time; (2) providing carriers and OEMs "a non-threatening solution for cross-vendor compatibility"; and (3) building a "community force around Google handset APIs and applications."[290]

The Android team thought a permissive open source license, such as Mozilla's, requiring licensees to maintain compatibility with Google APIs, was appropriate.[291] The team also saw Java as critical to their plan for numerous reasons: (1) "Carriers require it"; (2) "[Microsoft] will never do it"; (3) "Elegant tools story"; (4) "Safe sandbox for 3rd party developers"; (4) "Existing pool of developers and applications"; and (5) "Who pays? OEM pays [S]un a license, typically < .30 in volume."[292]

At the time, the Android team was planning to develop a clean room implementation of a Java virtual machine ("JVM").[293] They sought to obtain a Java™ logo certification for carrier certification, which would require a license from Sun. Their main concern was ensuring an open source JVM, not cost. The team proposed negotiating the first open source Java 2 Platform, Micro Edition JVM license with Sun.[294]

The Android team assumed they would be able to work out an open-source license with Sun.[295] By early October 2005, Rubin anticipated Sun would decline to collaborate on a joint project, but that Google could negotiate a license that granted rights to "open source" Android with Java APIs:

> We'll pay Sun for the license and the TCK [Technology Compatibility Kit]. Before we release our product to the open source community we'll make sure our JVM passes all TCK certification tests so

290. *See id.* at 6–7.

291. *See id.*

292. *See id.* at 8.

293. *See id.* at 9.

294. *See id.* The memo noted that Tim Lindholm, a former Sun Microsystems engineer who was involved with Java (*see* John Letzing, *Who Is Tim Lindholm? Google's CEO is Wondering That Too*, WALL ST. J. (Apr. 18, 2012), http://blogs.wsj.com/digits/2012/04/18/who-is-tim-lindholm-googles-ceo-is-wondering-that-too/ (last visited Jan. 27, 2018)), would lead the negotiation for Google, *see* Android GPS [Google Product Strategy], *supra* note 285, at 9. It was hoped that the negotiation would reinforce Google's JVM development or persuade Sun to open source its multiple virtual machine implementation. *See id.*

295. *See* Email from Andy Rubin at 14, 20–21 (Sept. 6, 2005), Trial Ex. 6, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012) (No. C 10-03561 WHA) (meeting notes from Aug. 30, 2005 Android GPS meeting; listing Java partnership as the first item on "Building Partnerships" slide (p.14); listing 4th quarter 2005 as milestone for Java partnership with Sun (p.21); estimating 4th quarter 2007 shipping date (p.20)).

> that we don't create fragmentation. Before a product gets brought to market a manufacturer will have to be a Sun licensee, pay appropriate royalties, and pass the TCK again.[296]

Rubin outlined two options if Sun declined: (1) "Abandon our work and adopt [Microsoft Common Language Runtime virtual machine] and C# language"; or (2) "Do Java anyway and defend our decision, perhaps making enemies along the way."[297]

As 2006 began, the Android team remained firmly committed to pursuing the Java API route and Sun appeared to be warming to a licensing agreement. Brian Swetland, an Android Senior Software Engineer, communicated that the team was "pretty set" on using Java and set forth a detailed set of reasons.[298] "[T]he negotiations with Sun are going far better than expected."[299] On January 13th, Rubin communicated to Sergey Brin the importance of Java for Android and explained he and Sun representatives had "conceptually agreed to open java and additionally to broaden the relationship" to create a Red Hat-type distribution model[300] with Sun for Android.[301] Rubin characterized the arrangement as an "industry changing partnership" which would lead Sun to "walk away from a $100M annual J2ME licensing business into an open source business model that we together crafted. This is a huge step for Sun, and very important for Android and Google."[302] By February, Scott McNealy, Sun's CEO, expressed enthusiasm to Eric Schmidt over jointly developing "an Open Source Java Linux Mobile Handset Platform implementation on the momentum of over 1 Billion Java Micro Edition based handsets deployed in the market currently."[303]

---

296. *See* Email from Rubin to Tracey Cole (Oct. 11, 2005), Trial Ex. 7, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012) (No. C 10-03561 WHA). Rubin had licensed Java for the Sidekick operating system, but that operating system did not substantially modify the platform. *See* VOGELSTEIN, *supra* note 270, at 57. The Android project, however, sought substantial modifications. Hence, the negotiations would be more difficult. *See id.*

297. *See* Email from Rubin to Tracey Cole (Oct. 11, 2005), *supra* note 296.

298. *See* Email from Brian Swetland (Jan. 2, 2006), Trial Ex. 13, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012) (No. C 10-03561 WHA).

299. *See id.*

300. *See Red Hat*, WIKIPEDIA, https://en.wikipedia.org/wiki/Red_Hat [https://perma.cc/94M5-HHQK].

301. *See* Email from Andy Rubin to Sergey Brin (Jan. 13, 2006), Doc. 398–10, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), http://www.fosspatents.com/2011/09/sun-proposed-red-hat-style-android.html [https://perma.cc/US4Q-K9SY].

302. *See id.*

303. *See* Email from Scott McNealy, contained in Email from Vineet Gupta (Feb. 9, 2006), Trial Ex. 16, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

In early March, however, McNealy expressed some reticence to Jonathan Schwartz: "The Google thing is really a pain. They are immune to copyright laws, good citizenship and dont [sic] share. They dont [sic] even call back."[304] Nonetheless, Rubin and Vineet Gupta, Sun's Chief Strategy/Technology Officer for OEM Software Systems Engineering, were deep into the process of marking up a draft Collaboration Development and License Agreement.[305]

In the midst of these negotiations, Jonathan Schwartz took over the CEO position from Sun co-founder McNealy.[306] The press reported that "McNealy and the company's employees and customers are all counting on Mr. Schwartz, a longtime admirer of Apple's co-founder, Steven P. Jobs, to find a way to recapture Sun's magic."[307] In taking the reins, Schwartz emphasized that Java was the number one driver of growth at Sun. "More teenagers recognize Java than they do Microsoft, because that is what they have in their pocket on their cellphone. Shame on me if I can't find a way to monetize that."[308]

During the intervening month, the push to create a Sun-Google collaboration lost momentum.[309] On April 28th, Rubin confidently emailed Alan Eustace, Senior Vice President of Engineering and Research at Google, and Schmidt: "I smell fear and think we're in a great negotiating position."[310] On the structure of the deal, Rubin summarized:

> 1) I am convinced they will open source java with no tricks
> 2) Final price: $28M
> 3) We did such a good [job] of convincing them our platform was a good idea, they want to have a hand in it's[sic] design and "own" parts where they have no value add.[311]

---

304. *See* Email from Scott McNealy (Mar. 8, 2006), Trial Ex. 563, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

305. *See* Email from Andy Rubin (Mar. 26, 2006), Trial Ex. 618, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (appending draft agreement and draft agreement with further mark-ups).

306. *See* John Markoff, *For Sun Microsystems, a Leader with Little Taste for Convention*, N.Y. TIMES (Apr. 26, 2006), http://www.nytimes.com/2006/04/26/technology/for-sun-microsystems-a-leader-with-little-taste-for-convention.html (last visited Jan. 27, 2018).

307. *See id.*

308. *See id.*

309. *See* Email thread from Gupta (May. 8, 2006), Trial Ex. 2372, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

310. *See* Rubin Email thread (Apr. 28, 2006), Trial Ex. 3443, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

311. *See id.*; *see also* Google's Trial Brief, No. 1706, at 3–4, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) ("By the end of April 2006, though other terms of their partnership remained unsettled, Sun had agreed to accept a

Rubin indicated he was not onboard with the third point. Schmidt replied the next day to say that he had not heard back from Schwartz and to remind Rubin to make sure that Larry Page was comfortable with the deal, noting that Page "is loathe [sic] to accept any restrictions on us."[312]

On May 4th, Rubin emailed Schwartz proposing a meeting "to hash this out and get the deal back on track . . . [F]rom the email exchange between you and Eric [Schmidt], it's obvious to me that both parties want to make this work. One final push may be all it takes."[313] The negotiations, however, soon hit an impasse over the code forking issue.[314]

Google opted for Plan B: "Do Java anyway and defend our decision." The Android team pushed ahead with its own Java implementation.[315] Using the Java language would not be a problem as Sun had released it to the public. But the Android team also wanted to use selected Java API packages from the Java Standard Edition and develop its own virtual machine.

If the Java programming language is analogized to the letters, words, and syntax of the English language, the API implementations can roughly be characterized as paragraphs or chapters within a book written in the Java language.[316] Copying the full API implementations, involving large chunks of code, would run afoul of copyright law. The Google team believed that Android could achieve its goals by emulating the API functionality with independently written implementing code. By avoiding Sun's restrictive licensing terms,

---

payment from Google of $28 million over three years to compensate Sun for the risk of lost licensing revenue that might result from an open source Android platform.").

312. *See* Andy Rubin Email thread (Apr. 28, 2006) Trial Ex. 3443, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

313. *See* Email thread from Vineet Gupta (May. 8, 2006) Trial Ex. 2372, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

314. *See* Email from Eric Schmidt to Andy Rubin (May 14, 2006), Trial Ex. 215, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA); Email from Desalvo to Rubin (Jun. 1, 2006), Trial Ex. 2372, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA); VOGELSTEIN, *supra* note 270, at 57 (reporting that Sun would not agree to forking of its platform); Email from Andy Rubin to Bob Lee (Aug. 11, 2007), Trial Ex. 230, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (explaining Sun's profit motivation for choosing GPL for Java ME: "Sun chose GPL . . . so that companies would need to come back to them and take a direct license and pay royalties."; and noting that Google "negotiated 9 months with Sun and decided to walk away after they threatened to sue us over patent violations.").

315. *See* Email from Chris Desalvo to Andy Rubin (Jun. 1, 2006), Trial Ex. 215, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) ("With talks with Sun broken off where does that leave us regarding Java class libraries? Ours are half-ass at best. We need another half of an ass.").

316. There are, however, critical limitations to this analogy for purposes of copyright analysis. API packages, unlike words, function as the gears and levers of a virtual machine. *See infra* notes 631–33.

Google could blaze its own trail without Sun's meddling.[317] Of particular importance, Google sought to avoid the GNU GPL to provide Android adopters — carriers, OEMs, chip-makers, and other component manufacturers — greater opportunity to customize and profit from their own innovations and market strategies. More permissive open licenses, such as the BSD, Mozilla, and Apache licenses, better fit Google's vision.

Google recognized that this path involved risk of copyright and patent liability. The copyright issue turned on whether and to what extent copyright law protected the function labels and structure, sequence, and organization ("SSO") of Java APIs. Because of the Supreme Court's deadlock in *Lotus v. Borland*, the First Circuit's treatment of function labels as uncopyrightable methods of operation strictly governed only in the First Circuit. Nonetheless, the Second Circuit's *Altai* decision and the Ninth Circuit's *Apple* decision exposed the weakness of the Third Circuit's superficial analysis of SSO in *Whelan*. Furthermore, the *Altai* decision and the Ninth Circuit's *Sega* decision clearly viewed achieving interoperability with another computer interface through a different implementation to be fair game. Yet Android was aiming for something other than complete end user interoperability. It wanted to pick and choose among interface elements in building a new platform with an optimized interface for a different consumer marketplace.

The Sun-Microsoft controversy further complicated the analysis. Microsoft had licensed Java and agreed not to fork the code.[318] When it did, Sun sued for breach of contract, copyright infringement, trademark infringement, and unfair competition.[319] Although Sun ultimately enjoined Microsoft's incompatible Java implementations and recovered $20 million in damages, the copyright issue was never squarely resolved in a judicial decision. The later antitrust settlement only further complicated the matter. Would Sun see Google's forking of the Java Standard Edition API as similarly anti-competitive?

The Google strategists faced serious legal and reputational risk proceeding without some sort of collaboration with Sun or a Java license.[320] But by not proceeding quickly and independently, Google

317. *See* Email from Tim Lindholm to Andy Rubin (Mar. 24, 2006), Trial Ex. 18, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (expressing consternation at Sun's licensing model: "Ha, wish them luck. Java.lang api's are copyrighted. And Sun gets to say who they license the tck [Technology Compatibility Kit used to ensure Java compatibility, *see* Appendix A] to, and forces you to take the 'shared part' which taints any clean room implementation.").

318. *See Fork (Software Development)*, *supra* note 16; *see also* Appendix A (defining forking).

319. *See supra* text accompanying notes 231–56.

320. *See* Email from Tim Lindholm to Andy Rubin (Oct. 26, 2005), Trial Ex. 125, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) ("If we don't show strong efforts toward avoiding fragmentation we are also going to have much

faced other risks to its core business as mobile computing emerged. The Microsoft and Symbian mobile platforms were gaining market share and Apple was poised (and rumored) to be entering the mobile computing marketplace.[321]

Over the next two years, the Android team independently developed its own implementing code for 37 of the 166 Java API packages in the Java Standard Edition[322] and an independent virtual machine ("Dalvik"). In this way, the Android operating system emulated the functionality of known and tested APIs that fit the Android team's constrained design parameters. The Android design effort can be analogized to the Sun Green Project team's adaptation of the C programming language to design a secure, reliable, object-oriented, platform-independent language that could interpret other languages and could function on small computer chips embedded in consumer devices.[323] It can also be analogized to their earlier effort to adapt Oak for the web, which resulted in Java.[324] Android's use of the same function labels as Java would enable millions of Java programmers to quickly master Android app development. Although Android apps would not be fully interoperable with Java, they were similar enough and better optimized to the constraints of mobile devices.[325] This clean room effort added substantially more time and cost to Android develop-

---

more trouble with Sun."); Email from Andy Rubin to Eric Schmidt (Nov. 14, 2007), Trial Ex. 180, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (commenting that the Java licensing issue "is a touchy subject").

321. *See Timeline of Apple "iPhone" Rumors (1999-present)*, FIERCE WIRELESS (Dec. 18, 2006 10:26 AM), http://www.fiercewireless.com/story/timeline-apple-iphone-rumors-1999-present [https://perma.cc/HY7C-QJSE].

322. *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp.2d 974, 977 (N.D. Cal. 2010), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014). Appendix A lists and summarizes the 37 APIs.

As a lead Android programmer would later explain:

> there's certain of these APIs which you . . . fundamentally think of as . . . part of the system that you can just use without really having to think too much about it. . . . [M]y job was . . . to . . . sift through all of that and come up with a nice and consistent set of APIs that we have would then implement and provide to developers.

See Testimony of Dan Bornstein, Trial Tr. at 1782–83, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA). The goal was not to implement all of the API packages present in any particular Java Platform, but rather "to provide something that was familiar to developers" in a "good mobile platform" that met "certain constraints" of that medium, such as battery limitations, less memory than a desktop computer or server, and slower CPU speed. *See id.* at 1783–84.

323. *See supra* text accompanying notes 196–205.

324. *See id.*

325. *See* Stephen Shankland, *Google Carves an Android Path Through Open-source World: Google Is Committed to Many Open-source Tenets With Its Android Mobile Phone Software — But it's Willing to Step on a Few Open-source Toes, Too*. CNET (May 22, 2008), http://www.cnet.com/news/google-carves-an-android-path-through-open-source-world/ (last visited Jan 27, 2018).

ment, but avoided literal copying of the Java API implementation code.[326]

Within the larger Google enterprise, the company hedged its mobile strategy by pursuing two paths: (1) working with Apple, which was developing a phone platform, to integrate Google applications; and (2) developing the independent Android platform. Rival groups within Google competed for primacy.[327] Even within the Android path, there was some tension about whether to focus on software (Schmidt's instinct) or develop a Google handset (Page's vision).[328] Google was a software company, with no experience in designing and manufacturing devices.

By the end of 2006, the Android team had been working intensively for the better part of two years developing code, negotiating license and partnership agreements, and designing prototypes. They were on track to release the Android platform by the end of 2007.[329] Those plans encountered a seismic jolt on January 9, 2007, the day Steve Jobs unveiled the iPhone to a rapturous response.[330] Rubin immediately realized that "we're not going to ship *that* [the current version of the Android] phone."[331] It looked conventional and lacked the magical touchscreen and seamless design of the iPhone. While the Android platform and phone was more advanced than the iPhone in many of its features and integration with Google web applications, it had nowhere near the visual and tactile appeal of the iPhone.[332]

After the initial shock of the iPhone announcement, the Android team realized that Apple's remarkable device and business plan played into Android's "open platform" strategy. Apple had entered into an exclusive distribution deal with AT&T, one of the major telcos.[333] The other telcos, some of whom had been hesitant to partner with Google, were now anxious to join forces to compete with AT&T.[334] Moreover, Apple's proprietary platform left little room for telcos to develop distinctive features. Android's open platform and more generous partnership terms provided greater opportunity for

---

326. *See* VOGELSTEIN, *supra* note 270, at 57 (reporting that "[w]ithout the Java code, Rubin had to spend months of extra time creating a work-around").

327. *See id.* at 62, 84–95.

328. *See id.* at 56–57.

329. *See id.* at 45.

330. *See* John Markoff, *Apple Introduces Innovative Cellphone*, N.Y. TIMES (Jan. 10, 2007), at A1, http://www.nytimes.com/2007/01/10/technology/10apple.html (last visited Jan. 27, 2018).

331. *See* VOGELSTEIN, *supra* note 270, at 46; *see also id.* at 45 (quoting Chris DeSalvo: "As a consumer I was blown away. I wanted on immediately. But as a Google engineer, I thought, 'we're going to have to start over.'").

332. *See id.* at 47.

333. *See* Markoff, *supra* note 330 (reporting that the iPhone would be available solely through Cingular Wireless, AT&T's wireless division, by mid-year).

334. *See* VOGELSTEIN, *supra* note 270, at 119–121.

telcos to differentiate their products, innovate, and profit.[335] Furthermore, Google's partnering with Apple on the iPhone through integration of Google applications and assurances from Google leaders that Android was not a significant initiative lulled Steve Jobs into a false sense of security that Google was not seriously pursuing a robust competing platform or line of products.[336]

The fanfare surrounding the iPhone announcement rallied support within Google for the Android project. Google's leadership came to see Apple's rapid rise in the mobile computing field as a threat to its core businesses in much the same way that Microsoft had dominated desktop computing.[337] Google allocated more resources to the Android project.[338] The Android team found negotiating partnerships with telcos and OEMs far easier.[339] By working around Sun on the Java API copyright issue, Android programmers had greater flexibility to optimize the platform without interference from Sun.[340] Google leadership pressured the Android team to accelerate Android's release.[341]

Google began the rollout of the Android platform in early November 2007.[342] On November 5th, Google unveiled the Open Hand-

---

335. *See id.* Google sweetened the partnership for telcos by offering them a cut of app revenues. This motivated the carriers to push Android phones, which in the end contributed to Google's bottom line through enhanced use of Google applications. The combined push catapulted Android to record sales. *See id.* at 123.

336. *See id.* at 84–103, 113–15, 129.

337. *See id.* at 129–30.

338. *See* Android GPS Meeting Notes (Jul. 17, 2007), Trial Ex. 433, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA); VOGELSTEIN, *supra* note 270, at 83–84.

339. *See* VOGELSTEIN, *supra* note 270, at 119–21.

340. *See* Email from Andy Rubin to Eric Schmidt (May 11, 2007), Trial Ex. 207, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (referring to Sun's renewed interest to discuss mobile technology and favoring independence:

> I don't see any way we can work together and not have it revert to arguments of control. I'm done with Sun (tail between my legs, you were right). They won't be happy when we release our stuff, but we now have a huge alignment with industry, and they are just beginning. While I'm not underestimating their abilities, when folks like DoCoMo [leading mobile phone operator in Japan] tell us they want to dump Sun for us, I'm assuming we have something valuable and good.

).

341. *See* Email from Eric Schmidt to Andy Rubin, Larry Page, Sergey Brin, et al. (Jan. 15, 2007), Trial Ex. 216, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) ("I'd like to have an Android GPS as soon as practical"); VOGELSTEIN, *supra* note 270, at 83.

342. *See Open Source Alliance, Industry Leaders Announce Open Platform for Mobile Devices: Group Pledges to Unleash Innovation for Mobile Users Worldwide*, OPEN HANDSET ALLIANCE (Nov. 5, 2007), http://www.openhandsetalliance.com/press_110507.html [https://perma.cc/DQ9V-GXT2]; Miguel Helft & John Markoff, *Google Enters the Wireless World*, N.Y. TIMES (Nov. 5, 2007), http://www.nytimes.com/2007/11/05/technology/05cnd-gphone.html (last visited Jan. 27, 2018); Saul Hanseel, *The*

set Alliance, a consortium of handset makers, application developers, telcos, and component manufacturers (such as chip makers), in conjunction with the outlines of the Android platform.[343] Andy Rubin explained that Android's software was based on the Linux operating system and Sun's Java language, which would enable programmers to easily develop applications that connect to independent Web services.[344]

Jonathan Schwartz, Sun's CEO, publicly applauded Google's use of Java, proclaiming that Google had "strapped another set of rockets to the [Java] community's momentum-and to the vision defining opportunity across our (and other) planets."[345] Privately, Sun feared that Android's use of Java would undermine its WORA paradigm and its mission to establish Java ME as the leading mobile platform and a significant revenue generator.[346] Following Google's November 5th Android announcement, Jonathan Schwartz communicated to colleagues that "[a] separate implementation isn't a fork — so long as Google agrees to certify their platform as compliant with the Java specification. If they don't, they won't be able to call it Java."[347] In an "off the record" communication with a New York Times reporter one day after the Android announcement, Schwartz sniped about Google's opposition to Sun's plan to open source Java.[348]

The Android announcement produced significant fallout beyond Sun. Steve Jobs saw the Android announcement as betrayal by Brin, Page, and Schmidt.[349] Schmidt had served on Apple's Board of Direc-

---

Gphone: So Open It Could Be Closed, N.Y. TIMES (Nov. 5, 2007), http://bits.blogs.nytimes.com/2007/11/05/the-gphone-so-open-it-could-be-closed/ [https://perma.cc/H2UX-9U2J].

343. *See Open Handset Alliance*, WIKIPEDIA, https://en.wikipedia.org/wiki/Open_Handset_Alliance [https://perma.cc/2YTZ-Z9ZJ].

344. *See* Miguel Helft & John Markoff, *Google Enters the Wireless World*, N.Y. TIMES (Nov. 5, 2007), http://www.nytimes.com/2007/11/05/technology/05cnd-gphone.html (last visited Jan. 27, 2018).

345. *See* Jonathan I. Schwartz, *Congratulations Google, Red Hat and the Java Community!*, JONATHAN'S BLOG! (Nov. 5, 2007), http://web.archive.org/web/20101023072550/http://blogs.sun.com/jonathan/entry/congratulations_google [https://perma.cc/53KC-GXBJ].

346. *See* Email thread involving Vineet Gupta (Sun) (Sep. 24, 2007), Trial Ex. 565, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

347. *See* Email from Schwartz (Nov. 12, 2007), Trial Ex. 1055, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA). At the time that Schwartz wrote that Email, Google had not yet released the Android SDK.

348. *See* Email from Jonathan Schwartz to John Markoff (Nov. 6, 2007), http://www.fosspatents.com/2012/04/former-sun-chief-about-google-immune-to.html [https://perma.cc/NRD5-QSFJ].

349. *See* WALTER ISAACSON, STEVE JOBS 511–14, 524, 563 (2011). After initially disbelieving that Google had betrayed him, *see id.* at 95, Steve Jobs declared war over the Android betrayal. Jobs characterized its 2011 patent infringement suit against HTC (and, by extension, Android) as saying:

> 'Google, you fucking ripped off the iPhone, wholesale ripped us off.' Grand theft. I will spend my last dying breath if I need to, and I will spend every penny of Apple's $40 billion in the bank, to right this

tors since 2006.[350] The ensuing jockeying for mobile phone patent portfolios, lawsuits, and interpersonal repercussions restructured major industries. The growing rift between Apple and Google generated rivalry with the iPhone and rallied support, even among those who had worked to support integration of Google applications with the iPhone, for a robust, independent, and competitive Android platform.[351]

Based on the Android SDK, Sun and other industry observers could see that Google was diverging from the Java standard platform and the Java Community Process.[352] Google deflected suggestions that Android fragmented Java by focusing attention on how the Open Handset Alliance provided a more responsive, less restrictive, open platform for mobile devices.[353] Sun and Google continued to monitor

---

> wrong. I'm going to destroy Android, because it's a stolen product. I'm willing to go thermonuclear war on this. They are scared to death, because they know they are guilty. Outside of Search, Google's products — Android, Google Docs — are shit.

Id. at 512.

350. *See* Dr. Eric Schmidt Resigns from Apple's Board of Directors, APPLE NEWSROOM (Aug. 3, 2009), https://www.apple.com/pr/library/2009/08/03Dr-Eric-Schmidt-Resigns-from-Apples-Board-of-Directors.html [https://perma.cc/P58N-LTT9] (quoting Steve Jobs:

> Eric has been an excellent Board member for Apple, investing his valuable time, talent, passion and wisdom to help make Apple successful. Unfortunately, as Google enters more of Apple's core businesses, with Android and now Chrome OS, Eric's effectiveness as an Apple Board member will be significantly diminished, since he will have to recuse himself from even larger portions of our meetings due to potential conflicts of interest. Therefore, we have mutually decided that now is the right time for Eric to resign his position on Apple's Board.

).

351. *See* VOGELSTEIN, *supra* note 270, at 115–19.

352. *See* Stephen Shankland, *Sun's Worried that Google Android Could Fracture: Java Company's Software Chief Wants to Work with Google to Make Sure that the Android Phone Software Won't Split Java into Incompatible Versions*, CNET (Nov. 14, 2007), http://www.cnet.com/news/suns-worried-that-google-android-could-fracture-java/ (last visited Jan. 27, 2018) [hereinafter Shankland, *Sun's Worried that Google Android Could Fracture*] (reporting that:

> [p]ainful flashbacks are beginning to torment those of us who lived through the Java wars between Sun Microsystems and Microsoft that began 10 years ago. Earlier this week, Google released programming tools for its Android mobile-phone software project that shun the existing Java standard-setting process in favor of a Google-specific variety. Sun responded on Wednesday by expressing concern that Google's Android project could fragment Java into incompatible versions.

); *see also* Stephen Shankland, *Google's Android Parts Ways with Java Industry Group Heads Up, Programmers: Google Opted to Create its Own Java Standards and Technology for its Android Mobile Phone, Not Piggyback on the Existing Java Community Process*, CNET (Nov. 13, 2007), http://www.cnet.com/news/googles-android-parts-ways-with-java-industry-group/ (last visited Jan. 27, 2018).

353. *See id.*; Shankland, *Sun's Worried that Google Android Could Fracture* (quoting a Google press statement:

each other's activities warily as Android products moved into the marketplace in 2008 and 2009,[354] a period in which Apple's iPhone was ascendant. Leaders at both companies occasionally broached licensing and collaboration,[355] but a gulf remained.[356] Sun refrained from blocking Android through legal action.

The marketplace quickly resolved the fate of the two companies. With Java ME failing to take off, Sun became an acquisition target.[357] Rubin's vision proved prescient: "'When you have multiple O.E.M.'s building multiple products in multiple product categories, it's just a matter of time' before sales of Android phones exceed the sales of

---

> Google and the other members of the Open Handset Alliance are working to help solve fragmentation and supporting the developer community by creating Android, a mobile platform that responds to the needs of the developers, has the backing of industry leaders, and will be available as open source under a nonrestrictive license.

).

354. *See* Email from Vineet Gupta to Jonathan Schwartz (Oct. 23, 2008), Trial Ex. 2070, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (indicating that Google's Android "proposal more than likely is going to be about buying out Java"); Email from Andy Rubin to Dick Wall (Mar. 24, 2008), Trial Ex. 29, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (warning Google representatives not to demonstrate Android features to Sun employees or lawyers at JavaOne convention); Email from Dave Sobata to Tim Lindholm (Feb. 19, 2009), Trial Ex. 326, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (raising the question of who will own Java if Sun collapses and suggesting Google could buy the patent and copyright rights as a way of making "[o]ur Java lawsuits go away"); Email from Tim Lindholm to Daniel Bornstein (Apr. 29, 2009), Trial Ex. 1029, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (recommending avoiding interaction with Sun so as to avoid "inadvertently stir[ring] anything up for Android").

355. *See* Lindholm-Rubin Email thread (Nov. 24, 2008), Trial Ex. 1002, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (discussing recent efforts by Sun to "certify Android through the Java process and become licensees of Java."); Email from Eric Schmidt to Jonathan Schwartz (Mar. 31, 2008), Trial Ex. 3466, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (Re: update on android licensing; "We are happy to have our team meet with anyone at Sun who would like more information or who has ideas for us"; calling attention to an explanation of why Google chose to distribute Android to the public using the Apache v2 license); *see also* Ryan Paul, *Why Google Chose the Apache Software License over GPLv2 for Android*, ARS TECHNICA (Nov. 6, 2007), http://arstechnica.com/uncategorized/2007/11/why-google-chose-the-apache-software-license-over-gplv2/ [https://perma.cc/U4HB-HW2C] (linked in Schmidt's March 31, 2008 Email to Schwartz).

356. Sun had proposed to license Java to Google for $60 million over three years plus an additional amount of up to $25 million per year in revenue sharing. *See* Letter from Scott Weingaertner (Counsel to Google) to Judge Alsup at 5, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), https://www.scribd.com/document/58133136/Oracle-Google-Damages-June-6-Precis-Unredacted (last visited Jan. 27, 2018). It is unclear whether that offer would have afforded Google the flexibility and independence in developing Android that it sought.

357. *See* Patrick Thibodeau and Elizabeth Montalbano, *Update: Oracle Buying Sun in $7.4B Deal*, COMPUTERWORLD (Apr. 20, 2009), http://www.computerworld.com/article/2523479/data-center/update--oracle-buying-sun-in--7-4b-deal.html (last visited Jan. 27, 2018).

proprietary systems like Apple's and R.I.M.'s."[358] Figure 2 tells the story. After a gradual start, Android took the global smartphone operating systems market by storm, surpassing 50% of global smartphone operating systems by the third quarter of 2011 and rising to 80% of the market by the middle of 2013. [359] It exceeded 84% of the market in 2016, with Apple's iOS coming in second place with about 15% of the market.[360]

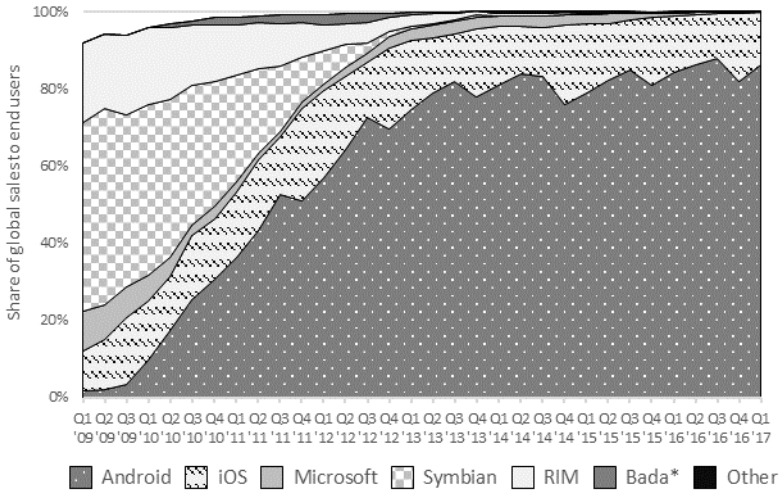

Figure 2. Global Market Share: Smartphone Operating Systems

3. Oracle's Acquisition of Sun Microsystems

Despite consternation over Android's "unofficial," non-standard, and incomplete Java implementation,[361] Sun declined to pursue legal

358. *See* Brad Stone, *Google's Andy Rubin on Everything Android*, N.Y. TIMES (Apr. 27, 2010), http://bits.blogs.nytimes.com/2010/04/27/googles-andy-rubin-on-everything-android/ [https://perma.cc/6RBL-HE7R].

359. *See* Statista, *Global Market Share Held By the Leading Smartphone Operating Systems in Sales to End Users from 1st Quarter 2009 to 1st Quarter 2016*, THE STATISTICS PORTAL (2016), http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ [https://perma.cc/W6CP-92XL?type=image].

360. *See id.*

361. *See* Dan Farber, *Java Creator James Gosling: 'Google Totally Slimed Sun,'* CNET (Apr. 30, 2012), http://www.cnet.com/news/java-creator-james-gosling-google-totally-slimed-sun/ [https://perma.cc/7MUC-UAY3] (quoting Gosling stating that Sun was "wronged" by Google and citing Sun's objections to Android's "very weak notions of interoperability" with Java); *Java (programming language)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Java_(programming_language) [https://perma.cc/X94V-5LG9] (referring to Android as an "unofficial" Java software platform); Joe Mullin, *Sun's Jonathan Schwartz at Trial: Java Was Free, Android Had No Licensing Problem*, ARS TECHNICA (May 11, 2016), http://arstechnica.com/tech-policy/2016/05/suns-jonathan-schwartz-at-trial-

action.[362] Such a course of action would have gone against Sun's long-standing cultural norms about open technology and evangelism within the industry.[363] Moreover, Sun could ill afford a prolonged litigation battle and the risk to Sun's reputation with other technology companies. Google was well-positioned financially and legally to put up a stiff defense. Sun's business was struggling and Wall Street and potential suitors would likely have seen such a lawsuit as a sign of desperation and a distraction from Sun's business goals.

With its hardware business in decline, software acquisitions sputtering,[364] and inability to monetize Java, Sun Microsystems's ability to move forward as an independent company came into question.[365] After acquisition negotiations with IBM failed in late 2008, Oracle successfully bid $7.4 billion in April 2009.[366] Oracle had built many of its software products with Java and hence had strong motivation to ensure that the Java platform would be in safe hands. Moreover, Oracle believed that it could significantly reduce Sun's operating costs as part of a combined company. It believed that the Sun products could bring in $1.5 billion in operating profits in the first year following the acquisition.[367]

Oracle's acquisition of Sun Microsystem dramatically altered the Java enforcement equation. Larry Ellison, Oracle's co-founder and CEO, had a reputation for brash business tactics.[368] Whereas Sun's leadership had embraced open technology with religious fervor, Ora-

---

java-was-free-android-had-no-licensing-problem/ [https://perma.cc/BZ28-SDF9] (quoting former Sun CEO expressing annoyance at Google's refusal to work out a license with Sun).

362. *See* Farber, *supra* note 361.

363. *See James Gosling: The Shit Finally Hits the Fan* . . . . (Aug. 12, 2010), http://news.java-virtual-machine.net/6018.html [https://perma.cc/T8EY-N5GV] (observing that "[f]iling patent suits was never in Sun's genetic code") (quoted in *Oracle's Java API Suit Against Google — Five Years Later*, FELDTHOUGHTS (Jun. 29, 2015), http://www.feld.com/archives/2015/06/oracles-java-api-suit-google-five-years-later.html [https://perma.cc/UQ8Q-CGKW]); Mullin, *Sun's Jonathan Schwartz at Trial*, *supra* note 178 (quoting Sun's CEO explaining that Android "was completely consistent with [Sun's] practices. When you say APIs are open, there are competitive implementations . . . It wasn't going to call itself Java, so there was nothing we could do"); *but see* Farber, *supra* note 361 (quoting Scott McNealy, Sun's co-founder and former CEO, disputing Schwartz's assertion that Sun allowed any forking of Java code so long as the implementer did not use the Java name or logo).

364. Sun had purchased StorageTek, a storage vendor, in 2005 for $4.1 billion and MySQL, a relational database company, in 2008, for $1 billion. *See* Jon Brodkin, *The Downfall of Sun Microsystems*, NETWORKWORLD (Apr. 24, 2009), http://www.networkworld.com/article/2268096/servers/the-downfall-of-sun-microsystems.html [https://perma.cc/XTP6-DCYM].

365. *See id.*

366. *See Oracle Buys Sun Microsystems for $7.4B*, CBS NEWS (Apr. 20, 2009), http://www.cbsnews.com/news/oracle-buys-sun-microsystems-for-74b/ [https://perma.cc/9YS8-QZLP] (reporting that analysts had long said that Sun could not stand on its own and were surprised when merger talks with IBM in late 2008 broke down).

367. *See* Brodkin, *supra* note 364.

368. *See supra* note 185.

cle's approach had been strategic. Unlike Sun, Oracle possessed the financial strength and diversified business strategy to pursue high stakes litigation. It had done well in recent years pursuing copyright litigation against SAP and instituting corporate takeovers.[369]

In announcing the Sun acquisition, Ellison characterized Java as "the single most important software asset we have ever acquired" and touted Oracle's Java-based middleware business, bolstered first by its BEA Systems acquisition[370] and purchase of Sun, as being "on track to become as large as Oracle's flagship database business."[371] Oracle would need to re-position Java's licensing business to achieve that goal. Oracle's leadership team sought to pursue a far more aggressive Java licensing strategy.

The Sun acquisition was completed in early 2010.[372] Oracle immediately approached Google about its use of Java in the Android platform. Google seriously considered alternatives to using Java,[373] but ultimately stood its ground because of the lack of good workarounds. For Oracle, the prospect of spending millions on attorneys' fees and costs for even a modest possibility of sharing in the large and growing Android marketplace was a plausible, if not attractive, business proposition. Moreover, it could quickly establish Oracle as a key player in the lucrative, strategically important, and rapidly growing mobile operating system marketplace. Delay would only enhance Google's laches and equitable estoppel defenses.

Yet Google would be a formidable adversary. Google was enormously profitable and had established a strong reputation for protect-

---

369. *See* Verne F. Kopytoff, *SAP Ordered to Pay Oracle $1.3 Billion*, N.Y. TIMES, Nov. 23, 2010; Jim Henschen, Oracle Lawsuit Against SAP Settled at Law, INFORMATIONWEEK (Nov. 14, 2016), http://www.informationweek.com/cloud/software-as-a-service/oracle-lawsuit-against-sap-settled-at-last/d/d-id/1317483 [https://perma.cc/R5NR-EUUD]; *Oracle Corp. v. SAP AG*, WIKIPEDIA, https://en.wikipedia.org/wiki/Oracle_Corp._v._SAP_AG [https://perma.cc/R5KF-BLJX]; *PeopleSoft*, WIKIPEDIA, https://en.wikipedia.org/wiki/PeopleSoft [https://perma.cc/7Z2Y-R8ZP].

370. *See* Larry Dugan, *Surprise! Oracle buys BEA Systems*, ZDNET (Jan. 16, 2008), http://www.zdnet.com/article/surprise-oracle-buys-bea-systems/ [https://perma.cc/YV8N-5BDB]. BEA Systems specializes in enterprise infrastructure software products.

371. *See* Patrick Thibodeau and Elizabeth Montalbano, *Update: Oracle Buying Sun in $7.4B Deal*, COMPUTERWORLD (Apr. 20, 2009), http://www.computerworld.com/article/2523479/data-center/update--oracle-buying-sun-in--7-4b-deal.html [https://perma.cc/X9LG-NLA7].

372. Antitrust authorities in the U.S. and Europe delayed the acquisition out of concern that Oracle, the leading relational database vendor, was acquiring a promising competing business (MySQL). *See* James Kanter, *New Snag for Oracle in Sun Deal*, N.Y. TIMES (Sept. 3, 2009), http://www.nytimes.com/2009/09/04/technology/companies/04oracle.html (last visited Jan. 27, 2018).

373. *See* Email from Tim Lindholm to Andy Rubin (Aug. 6, 2010), http://www.fosspatents.com/2011/11/googles-five-failed-attempts-to-give.html [https://perma.cc/EY8Y-KMSW] (noting that Page and Brin had asked engineers to "investigate what technical alternatives exist to Java for Android and Chrome. We've been over a bunch of these, and think they all suck. We conclude that we need to negotiate a license for Java under the terms we need.").

ing its business initiatives at substantial cost and with almost religious fervor. By mid-2010, Android had already surpassed Apple's market share of the global smartphone marketplace.[374] Google had fought long and hard to secure its core business assets and there was little reason to believe that its approach to defending Android would be any different. Google was actively defending patent lawsuits as well as copyright threats to YouTube and Google Books.[375] The conditions were set for a second API intellectual property battle royale.

B. The Oracle v. Google *Litigation*

After six months of negotiations with Google, Oracle fired a broadside salvo in the Northern District of California in August 2010, alleging that Android infringed Java-related patents and copyrights. With billions of dollars and control of two of the most important software platforms at stake, the parties would spare no expense in litigating the case over the next eight years, with more battles yet to unfold.

As background for understanding the complex issues surrounding legal protection for APIs, this Section chronicles the *Oracle v. Google* litigation. The key phases are: (1) the complaint; (2) the first trial followed by Judge Alsup's ruling that the Java APIs are not copyrightable; (3) the Federal Circuit's reversal of Judge Alsup's copyrightability ruling and remand for a fair use trial; (4) the interlocutory certiorari petition; (5) the fair use trial; and (6) the road ahead. Section III.C examines the uncertain copyright status of APIs. Part IV examines the district court and Federal Circuit decisions and assesses the larger policy ramifications.

1. Oracle's Complaint and Pretrial Case Management

Oracle's initial complaint alleged, in the barest of bones, that Android infringed seven utility patents and copyrights in the "code, documentation, specifications, libraries, and other materials that comprise the Java platform."[376] Oracle sought a permanent injunction and damages. The case was assigned to Judge William Alsup, an experienced

374. *See* Figure 2.

375. *See* Viacom International, Inc. v. YouTube, Inc., Civil Action No. 07 CV 2103 (S.D.N.Y. filed Mar. 13, 2007); Author's Guild, et al. v. Google Inc., Class Action Complaint, Civil Action No. 05 CV 8138 (S.D.N.Y. filed Sep. 20, 2005).

376. *See* Complaint for Patent and Copyright Infringement, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1 [https://perma.cc/QV4W-6KST].

and well-respected jurist who was not afraid of technologically complex subject matter.[377]

After Google challenged the adequacy of Oracle's copyright infringement allegations, Oracle asserted that:

> [a]pproximately one third of Android's Application Programmer Interface (API) packages . . . are derivative of Oracle America's copyrighted Java API packages . . . and corresponding documents. The infringed elements of Oracle America's copyrighted work include Java method and class names, definitions, organization, and parameters; the structure, organization and content of Java class libraries; and the content and organization of Java's documentation.[378]

Much of the pretrial case management revolved around the patent allegations, damages experts, admissibility of the August 2010 Lindholm Email,[379] and court-ordered mediation.[380] Google sought reexamination of the asserted patents in February 2011.[381] The PTO's

---

377. *See* Dan Farber, *Judge William Alsup: Master of the Court and Java*, CNET (May 31, 2012), http://www.cnet.com/news/judge-william-alsup-master-of-the-court-and-java/ (last visited Jan. 27, 2018).

378. *See* Amended Complaint for Patent and Copyright Infringement at ¶ 40, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/36 [https://perma.cc/Z5RA-G4MP].

379. *See* Email from Tim Lindholm to Andy Rubin (Aug. 6, 2010), *supra* note 373 (stating that:

> What we've actually been asked to do (by Larry and Sergei [sic]) is to investigate what technical alternatives exist to Java for Android and Chrome. We've been over a bunch of these, and think they all suck. We conclude that we need to negotiate a license for Java under the terms we need.

); *Failed attempt #7: Federal Circuit Denies Google Petition to Exclude Lindholm Email*, FOSS PATENTS (Feb. 6, 2012), http://www.fosspatents.com/2012/02/failed-attempt-7-federal-circuit-denies.html [https://perma.cc/Q4LB-X9KV]; *Google's Five Failed Attempts to Give Confidential Status to 'Damning' Email in Oracle Case*, FOSS PATENTS (Nov. 9, 2011), http://www.fosspatents.com/2011/11/googles-five-failed-attempts-to-give.html [https://perma.cc/P2GZ-8J5P].

380. *See* Order Re: Further Settlement Conferences, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (Mag. Judge Paul Grewal), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/848 [https://perma.cc/5DGS-HJKP] (stating:

> We are referred to as trial courts because, in the end, some cases just need to be tried. [¶] This case is a good example of why that is so. Despite their diligent efforts and those of their able counsel, the parties have reached an irreconcilable impasse in their settlement discussions with the undersigned.

) (emphasis in original).

381. *See* Darryl K. Taft, *Google Asks Patent Office for Second Opinion on Oracle's Android Claims*, EWEEK (Feb. 17, 2011), http://www.eweek.com/c/a/Application-

rejection of all claims in several of the Oracle patents,[382] although still subject to further review and appeal, provided Google with leverage to narrow the scope of the patent case or to stay part of the litigation. Under pressure from Judge Alsup, who sought to avoid multiple proceedings, Oracle dismissed many of its patent claims to get an earlier trial date.[383]

Google sought summary judgment on the copyright cause of action.[384] On September 15, 2011, Judge Alsup largely rejected Google's copyright summary judgment motion.[385] While agreeing with Google that "the names of the Java language API files, packages, classes, and methods are not protectable as a matter of law"[386] under the copyright doctrine which denies protection for names and short phrases,[387] the court nonetheless rejected Google's broader argument that API declarations (beyond short phrases) and documentation are unprotectable under the *scènes à faire*, merger, or methods of operation (§ 102(b)) doctrines. Judge Alsup concluded that Google's categorical approach "ignores the possibility that some method declarations (for example) may be subject to the merger doctrine or may be *scènes à faire*, whereas other method declarations may be creative contributions subject to copyright protection."[388] As for the methods of operation, Judge Alsup explained that "[e]ven if Google can show that *APIs* are methods of operation not subject to copyright

---

Development/Google-Asks-Patent-Office-for-Second-Opinion-on-Oracles-Android-Claims-100246 [https://perma.cc/BA4N-L8H2].

382. *See* Scott Daniels, *An Update on Oracle's Infringement Case Against Google*, USPTO LITIGATION ALERT™ (Feb. 14, 2012), http://blog.whda.com/2012/02/an-update-on-oracles-infringement-case-against-google/ [https://perma.cc/EJ39-6DB3].

383. *See Oracle-Google Trial to Start on April 16, 2012*, FOSS PATENTS (Mar. 13, 2012); *Oracle Offers Withdrawal of Three More Patents in Exchange for Spring Trial Against Google*, FOSS PATENTS (Mar. 9, 2012), http://www.fosspatents.com/2012/03/oracle-offers-withdrawal-of-three-more.html [https://perma.cc/G4AG-4KCC]; *Pressure Mounting on Oracle to Drop Patent Claims Against Google and Focus on Copyright*, FOSS PATENTS (Mar. 5, 2012), http://www.fosspatents.com/2012/03/pressure-mounting-on-oracle-
to-drop.html [https://perma.cc/J3XV-BKRZ].

384. *See* Mot. for Summary Judgment on Count VIII of Plaintiff Oracle Am.'s Amended Complaint filed by Google Inc., Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

385. *See* Order Partially Granting And Partially Denying Defendant's Mot. For Summary Judgment On Copyright Claim, Oracle Am., Inc. v. Google Inc., 810 F. Supp. 2d 1002 (N.D. Cal. 2011).

386. *Id.* at 1009–10.

387. *See* Material Not Subject to Copyright, 37 C.F.R. § 202.1(a) (2014) (Copyright Office regulation denying copyright registration for "Words and short phrases such as names, titles, and slogans"); Planesi v. Peters, No. 04-16936, slip op. at *1 (9th Cir. Aug. 15, 2005); Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) ("Sega's security code is of such de minimis length that it is probably unprotected under the words and short phrases doctrine.").

388. *See Oracle Am.*, 810 F. Supp.2d at 1010–11.

protection, that would not defeat Oracle's infringement claim concerning the accused *specifications*."[389]

After some wrangling, Judge Alsup established an April 2012 trial date.[390] He structured the trial in three phases: (I) copyright infringement claims; (II) patent infringement claims; and (III) all remaining issues, including damages and willfulness, if necessary.[391]

As the case wended its way toward trial, the core copyright allegations were boiled down to the following: (1) "12 Android files of source code (copied from 11 Java files), including rangeCheck"; (2) "Plain English descriptions in the user manual, sometimes called the API 'specifications'"; (3) "37 APIs but only as to their specific selection, structure, and organization, it being conceded that the implementing code is different"; and (4) "Android's entire source code and object code as derivative works of the 37 Java APIs."[392] The following elements or works were not at issue: (a) "Android's use of the Java programming language (other than any direct copying of source code)"; (b) "The titles and names of APIs, including all package and class names and definitions, fields, methods and method signatures (names in the left column of specifications)"; (c) "The idea of APIs"; and (d) "The Dalvik virtual machine."[393]

The parties agreed that Judge Alsup would decide the copyrightability of the Java APIs and the jury would decide copyright infringement, fair use, and whether any copying was *de minimis*.[394] Thus, the most salient copyright issue — the copyrightability of APIs — was not going to be tried to the jury.

2. 2012 Trial

The Oracle-Google trial opened to great fanfare in the technology and business communities. The case represented one of the major battlefronts in the rapidly developing "smartphone war." Just as the Oracle case was heading to trial, Google was engaged in other high stakes patent battles with smartphone patent owners.[395]

---

389. *See id.* at 1011 (emphasis in original).

390. *See* Order Setting Trial Date of April 16, 2012, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

391. *See* Final Pretrial Order, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA).

392. *See* Request for Statement of Issues Re Copyright, at 1–2, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/854 [https://perma.cc/GBP3-YV9T].

393. *See id.* at 2.

394. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2011).

395. In August 2011, Google announced its acquisition of Motorola Mobility. Motorola Mobility owned more than 17,000 patents (as well as another 7,500 patent applications) which Google believed would bolster Android's ability to survive the smartphone patent

Oracle emphasized three themes during the copyright phase of the trial: (1) that the Google engineers believed that they needed a Java license to develop the Android platform;[396] (2) the importance of the Java "Write Once, Run Anywhere" philosophy;[397] and (3) that designing APIs and writing its code is a highly creative activity.[398] Google countered with the following arguments: (1) Sun freely licensed the Java language, encouraged the use of the Java APIs (thereby leading software developers to believe that they were also freely available), and publicly welcomed and supported Android's use of Java;[399] (2) after Sun failed to build a successful Java phone or mobile platform, Oracle acquired Sun with the intention of shaking Google down for a share of Android's profits;[400] (3) Google independently implemented the functions of the Java 37 APIs at issue and, in any case, the Java API declarations are but a small portion of Android's 15 million lines of code;[401] and (4) Google made fair use of Java APIs.[402]

As a result of Judge Alsup's case management decision to reserve the copyrightability of APIs, the jury's infringement verdict was largely a foregone conclusion. Judge Alsup instructed the jury that Oracle's Java-related copyrights "cover the structure, sequence and organization [SSO] of the compilable code"[403] and that Google "agrees that the structure, sequence and organization of the 37 accused API packages in Android is substantially the same as the structure, sequence and organization of the corresponding 37 API packages in Java."[404] Judge Alsup further instructed the jury that "[w]hile indi-

---

arms race. *See* David Goldman, *Google Seals $13 Billion Motorola Buy*, CNN MONEY (May 22, 2012), http://money.cnn.com/2012/05/22/technology/google-motorola/ [https://perma.cc/EFN9-9T7B].

396. Oracle's lead counsel began the opening argument by quoting Tim Lindholm's August 6, 2010 Email to Andy Rubin:

> What we have actually been asked to do by Larry and Sergey is to investigate what technical alternatives exist to Java for Android. We have been over a bunch of these and think they all suck. We conclude that we need to negotiate a license for Java under the terms we need.

See Trial Tr. at 182–83, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 942); *see also id.* at 190–93 (quoting Google engineer Emails discussing Java licensing).

397. *See* Trial Tr. at 193–97, 209–10, 219–20, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 942).

398. *See id.* at 197–99, 213; *id.* at 831 (Google engineer who formerly worked at Sun acknowledging that there can be "creativity and artistry" in even a single method declaration).

399. *See id.* at 243–45, 247–53, 266–69.

400. *See id.* at 245–46, 269–70.

401. *See id.* at 258–59.

402. *See id.* at 247, 270–74.

403. *See* Final Charge To The Jury (Phase One) And Special Verdict Form at 8, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1018), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1018 [https://perma.cc/9338-9LHH].

404. *See id.* at 10.

vidual names are not protectable on a standalone basis, names must necessarily be used as part of the structure, sequence, and organization and are to that extent protectable by copyright."[405]

Oracle's principal copyright infringement argument boiled down to showing the jury a side-by-side comparison of Java and Android source code. As Figure 3 from Oracle's closing argument slide deck shows, Google conceded that it copied the API declarations.

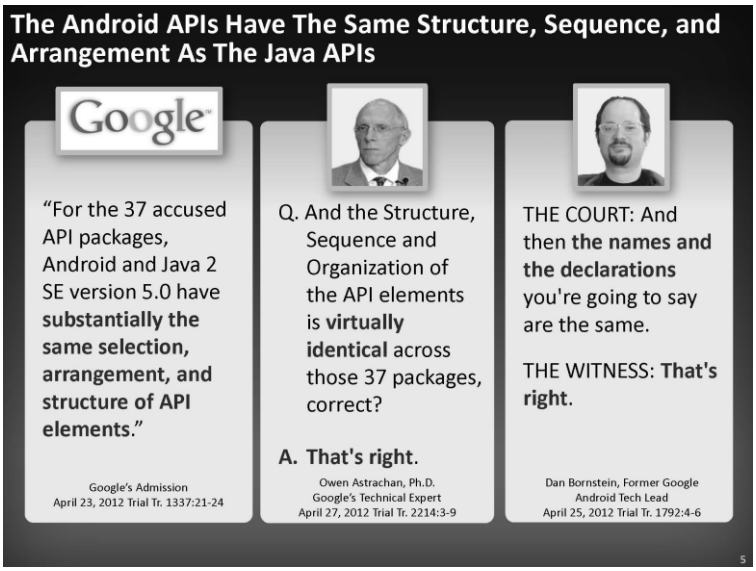

Figure 3. Oracle's Closing Argument Slide Deck, Slide 5
Google's Admission of Copying of Declarations

Oracle illustrated the copying of declarations with a side-by-side code comparison of one method (ClassLoader) from one class (Protection Domain) from the java.security API package.

405. *See id.*

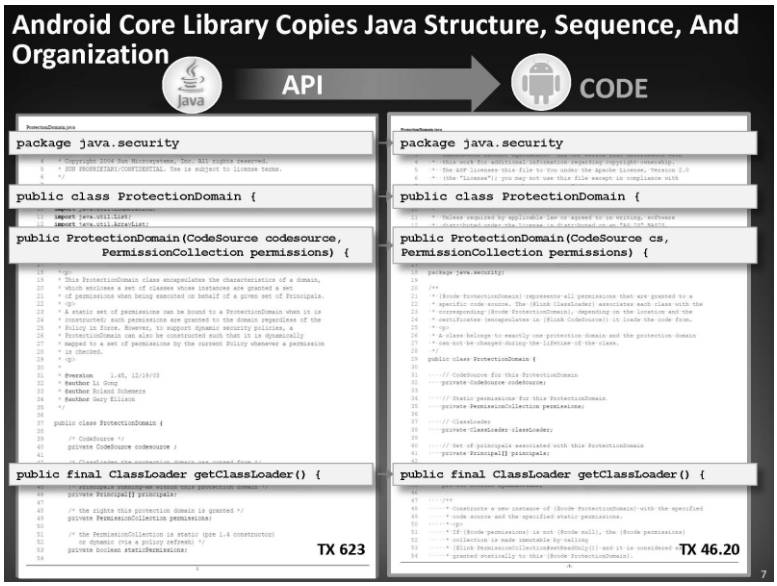

Figure 4. Oracle's Closing Argument Slide Deck, Slide 7
java.security ProtectionDomain ClassLoader

Oracle illustrated the extent of copying by showing the number of classes, methods, and declarations copied into Android.

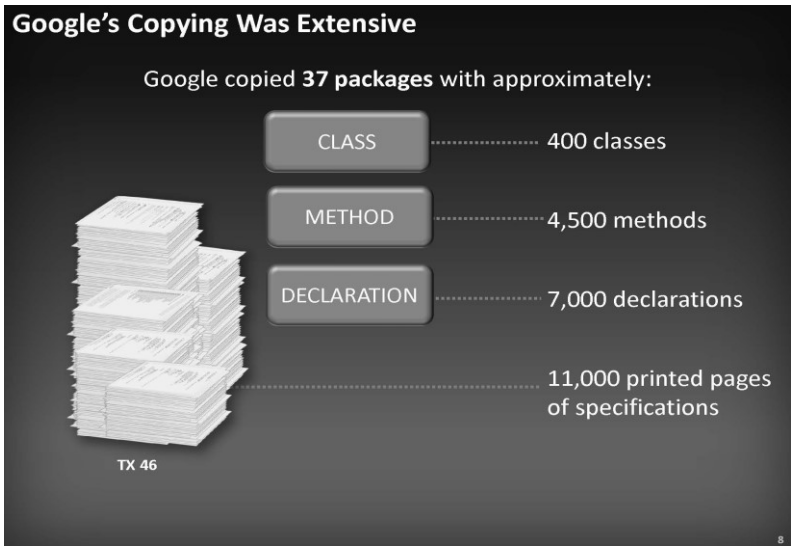

Figure 5. Oracle's Closing Argument Slide Deck,
Slide 8 on Extent of Copying

Beyond its motion seeking a determination that the Java APIs are not copyrightable,[406] Google's principal path to a trial victory was that the jury would find that Android's use of Java was permissible under the fair use doctrine. The jury would also provide factual input for Judge Alsup's assessment of equitable estoppel.

As the copyright phase of the trial wound down, the parties filed motions for judgment as a matter of law on all of the issues being litigated.[407] In an effort to focus on the key question, Judge Alsup requested that the parties answer sixteen questions regarding copyrightability of the structure, sequence, and organization of the APIs.[408]

Jury deliberations following the copyright phase of the trial ended with a partial Oracle victory.[409] Not surprisingly given Judge Alsup's API SSO instruction, the jury concluded that Android infringed the 37 Java API packages in question taken as a group.[410] The jury nonetheless held that Google did not infringe the documentation of the 37 Java API packages taken as a group under a virtual identity standard[411] and that the copying of eight of the nine specific source code

---

406. *See* Google's Mot. for Judgment as a Matter of Law on Sections Court VIII of Oracle's Amended Complaint, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 984), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/984 [https://perma.cc/7EBV-JSPM].

407. *See id.*; Oracle Am., Inc.'s Corrected Rule 50(A) Mot. at the Close of All Evidence, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1045), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1045 [https://perma.cc/UWF4-QVNT].

408. *See* Request for Further Phase One Briefing Re Copyrightability of SSO, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1057), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1057 [https://perma.cc/8NHB-SNJZ]; *see also* FURTHER ITEMS TO BRIEF IN TWENTY-PAGE BRIEFS DUE MAY 10, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1062), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1062 [https://perma.cc/2LLK-BNHL]; FURTHER ITEM FOR TWENTY-PAGE BRIEFS DUE MAY 10, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1088), https://docs.justia.com/cases/federal/districtcourts/california/candce/3:2010cv03561/231846/1088 [https://perma.cc/U8DN-LZVC].

409. *See* Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1089), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089 [https://perma.cc/V64F-ALYB]; Joe Mullin, *Google Guilty of Infringement in Oracle Trial; Future Legal Headaches Loom*, ARS TECHNICA (May 7, 2012), http://arstechnica.com/tech-policy/2012/05/jury-rules-google-violated-copyright-law-google-moves-for-mistrial/ [https://perma.cc/GJ5L-DK6J].

410. *See* Special Verdict Form at 1, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1089), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089 [https://perma.cc/V64F-ALYB].

411. *See* Final Charge To The Jury (Phase One) And Special Verdict Form at 12, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA)

files at issue was *de minimis*.[412] The jury hung on whether Google's infringement of the Java API SSO constituted fair use.[413] The jury further split on the special interrogatories relating to Google's equitable estoppel defense, holding that Sun/Oracle engaged in conduct that they knew or should have known would reasonably lead Google to believe that it would not need a license to use the Java API SSO, but that Google had not proven that it reasonably relied on such conduct.[414]

The patent phase of the trial commenced shortly after the jury rendered its copyright verdict. The same jury ruled that Google did not infringe the seven asserted claims of the two patents at issue.[415] Therefore, the need for a third phase of the trial hinged on Judge Alsup's resolution of the post-trial copyright motions.

One week later, Judge Alsup filed a released opinion holding that the Java APIs were not copyrightable.[416] This determination resulted in dismissal of the case. Although Judge Alsup cautioned that the ruling did not hold that "Java API packages are free for all to use without license" or that "the structure, sequence and organization of all computer programs may be stolen," the court ruled that "on the specific facts of this case the particular elements replicated by Google were free for all to use under the Copyright Act."[417]

Judge Alsup grounded his decision in the uncopyrightability of collections of functional attributes contained in the 37 Java APIs at issue and the fact that Google wrote its own implementing code.[418] The principal copying concerned the lines of declaring code, which

---

(ECF No. 1018), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1018 [https://perma.cc/R4CR-E4YL].

412. *See* Special Verdict Form at 2, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1089), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1089 [https://perma.cc/V64F-ALYB].

413. *See id.* at 1.

414. *See id.* at 3.

415. *See* Special Verdict Form, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (ECF No. 1190), https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1190 [https://perma.cc/6SPL-4RXP]; Josh Lowensohn, *Jury Verdict: Android Doesn't Infringe Oracle's Patents*, CNET (May 23, 2012), http://www.cnet.com/news/jury-verdict-android-doesnt-infringe-oracles-patents/ (last visited Jan. 27, 2018).

416. *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp.2d 974 (N.D. Cal. 2012). In a pyrrhic victory for Oracle, Judge Alsup granted judgment as a matter of law holding that Google's copying of the eight test files that the jury deemed de minimis were infringing. *See id.* at n.1.

417. *Id.* at 1002.

418. Google included a small (9 lines of a 3,179 line function), "innocent," and "inconsequential" segment of code (rangeCheck) in Android and eight test files that were never introduced into Android. *See id.* at 982–83. To clear the way for appeal, however, the parties stipulated, that these relatively modest code portions produced no damages. *See* Final Judgment, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA), (ECF No. 1211).

are necessary to operate the particular methods of the APIs at issue. As Judge Alsup explained:

> Significantly, the rules of Java dictate the precise form of certain necessary lines of code called declarations, whose precise and necessary form explains why Android and Java *must be* identical when it comes to those particular lines of code. That is, since there is only one way to declare a given method functionality, everyone using that function must write that specific line of code in the same way.[419]

While acknowledging that the overall structure of the Java API packages is creative, original, and "resembles a taxonomy," Judge Alsup nonetheless concluded that it functions as "a command structure, a system or method of operation — a long hierarchy of over six thousand commands to carry out pre-assigned functions."[420] Judge Alsup placed particular emphasis on *Sega* for its rejection of the Third Circuit's broad protection for the SSO of computer software[421] and its recognition that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright. 17 U.S.C. § 102(b)."[422]

Applying copyright's limiting doctrines as interpreted by Ninth Circuit cases[423] and following CONTU's guidance that when specific computer instructions, "*even though previously copyrighted, are the only and essential means of accomplishing a given task, their later*

---

419. *Oracle Am.*, 872 F. Supp. 2d at 979 (emphasis in original). *See id.* at 981 (finding that "[i]n order to declare a particular *functionality*, the [Java] language *demands* that the method declaration take a particular form (emphasis in original)); *id.* at 982 (finding that "the names of the methods and the way in which the methods are grouped" have to be the same in order to "be interoperable. Specifically, code written for one API would not run on an API organized differently, for the name structure itself dictates the precise form of command to call up any given method.").

420. *Id.* at 999–1000.

421. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1524–25 (9th Cir. 1992) (noting that "[t]he *Whelan* rule . . . has been widely — and soundly — criticized as simplistic and overbroad" (citing the Second Circuit's decision in *Computer Associates, Inc. v. Altai*)).

422. *See id.* at 1522.

423. The Ninth Circuit expressly endorsed the Second Circuit's *Altai* approach:

> Under a test that breaks down a computer program into its component subroutines and sub-subroutines and then identifies the idea or core functional element of each, such as the test recently adopted by the Second Circuit in *CAI*, 23 U.S.P.Q.2d at 1252–53, *many aspects of the program are not protected by copyright*. In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one.

Sega, 977 F.2d at 1525 (emphasis added).

use by another will not amount to an infringement,"[424] Judge Alsup determined that Google was free to write code that accomplished the same functionality as the Java APIs at issue even if it did not achieve complete compatibility with the full Java platform:

> While fragmentation is a legitimate business consideration, it begs the question whether or not a license was required in the first place to replicate some or all of the command structure. (This is especially so inasmuch as Android has not carried the Java trademark, and Google has not held out Android as fully compatible.) The immediate point is this: fragmentation, imperfect interoperability, and Oracle's angst over it illustrate the character of the command structure as a functional system or method of operation.[425]

In essence, later developers can achieve the *particular functionality* or method of operation of an API subsystem (and even groups of subsystems) so long as they write their own code and that method is not protected by a patent.

Judge Alsup's framework provided a general and concrete solution to the API copyright puzzle. Although he cautioned that his opinion was limited to the facts of the case and did not declare APIs uncopyrightable, Judge Alsup's analysis illuminated a clear pathway for software developers seeking to use APIs defined and first implemented by others without running afoul of copyright law.[426] Later developers are free to use declaring code so long as they use a clean room to implement the declarations. To many in the software industry, the ruling validated what was considered a best practice.[427] To others, it jeopardized the substantial effort and investment in developing software platforms and pioneering products, and threatened to undermine interoperability.[428]

---

424. *Oracle Am.*, 872 F. Supp. 2d at 986 (quoting CONTU REPORT, *supra* note 47, at 20 (emphasis added by Judge Alsup)).

425. *Oracle Am.*, 872 F. Supp. 2d at 1000.

426. Patent protection, trade secret law, and contractual limitations could nonetheless stand in the way, but copyright protection could not bar re-implementation of functional features of computer programs.

427. *See* Nick Wingfield & Quentin Hardy, *Google Prevails as Jury Rebuffs Oracle in Code Copyright Case*, N.Y. TIMES (May 26, 2016), http://www.nytimes.com/2016/05/27/technology/google-oracle-copyright-code.html (last visited Jan. 23, 2018) (quoting representatives of the Electronic Frontier Foundation, Public Knowledge, and a venture capital firm praising the jury's verdict); *supra* text accompanying notes 176–77.

428. *See* Annette Hurst, *Op-ed: Oracle Attorney Says Google's Court Victory Might Kill the GPL*, ARS TECHNICA (May 27, 2016), http://arstechnica.com/tech-policy/2016/05/op-ed-oracle-attorney-says-googles-court-victory-might-kill-the-gpl/ [https://perma.cc/66KG-2Z4C]; Florian Mueller, *Google's 'Fair Use' Defense Against Oracle Is an Insult to Human Intelligence: Android's Use of Java APIs Violates Copyright*, FOSS PATENTS (May 22,

3. Federal Circuit Appeal

Oracle filed its appeal with the U.S. Court of Appeals for the Federal Circuit.[429] Regional circuit law binds the Federal Circuit when reviewing questions of law and precedent not exclusively assigned to the Federal Circuit.[430] Thus, the Federal Circuit was required to review the copyright issues according to Ninth Circuit precedents.[431]

The appeal attracted broad interest in the technology sector, with established software companies favoring Oracle[432] and start-ups and application developers favoring Google on the API copyrightability issue.[433] Among the more notable briefs was the one filed by former Sun executives Scott McNealy and Brian Sutphin.[434] They emphasized the creativity involved in API design.[435]

Picking up on that theme, Oracle began its brief with a creative literary analogy:

> Ann Droid wants to publish a bestseller. So she sits down with an advance copy of HARRY POTTER AND THE ORDER OF THE PHOENIX — the fifth book — and proceeds to transcribe. She verbatim copies all the chapter titles — from Chapter 1 ('Dudley Demented') to Chapter 38 ('The Second War

---

2016), http://www.fosspatents.com/2016/05/googles-fair-use-defense-against-oracle.html [https://perma.cc/J79U-4QA3].

429. The Federal Circuit has exclusive jurisdiction over appeals from district court cases involving patent infringement allegations even though, as was the circumstance in *Oracle v. Google*, neither party challenged the district court's patent rulings.

430. *See* Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990).

431. Copyright issues are not exclusively assigned to the Federal Circuit. *See* 28 U.S.C. § 1295 (2012).

432. The Business Software Alliance, one of the largest and oldest software trade associations, as well as Microsoft Corp. and other established companies favored Oracle. *See* Corrected Brief for BSA | the Software Alliance as Amicus Curiae in Support of Plaintiff-Appellant Oracle Am., Inc., Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022 (Fed. Cir. Feb. 22, 2013); Brief for Amici Curiae Microsoft Corporation, EMC Corporation, and Netapp, Inc. in Support of Appellant, Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022 (Fed. Cir. Feb. 19, 2013).

433. *See* Brief of Amici Curiae Rackspace US, Inc., Application Developers Alliance, TMSOFT, LLC, and Stack Exchange Inc., Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022) (Fed. Cir. May 30, 2013); Corrected Brief of Amici Curiae of Software Innovators, Start-ups, and Investors in Support of Affirmance, Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022 (Fed. Cir. May 30, 2013).

434. *See* Corrected Brief of Scott McNealy & Brian Sutphin as Amici Curiae in Support of Reversal, Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022 (Fed. Cir. Feb. 22, 2013)

435. *See id.* at 8 ("Java's success rested in large part upon its elegant and creative set of packages that Sun designed and developed . . . . [T]hese packages provide a lengthy and creative set of pre-existing programs that made it much easier for Java programmers to quickly write programs and intuitively grasp and learn the Java platform."); *id.* at 13 ("The Selection Naming and Organization of Java's Packages (APIs) Are Unique and Creative.").

> Begins'). She copies verbatim the topic sentences of each paragraph, starting from the first (highly descriptive) one and continuing, in order, to the last, simple one ('Harry nodded.'). She then paraphrases the rest of each paragraph. She rushes the competing version to press before the original under the title: Ann Droid's HARRY POTTER 5.0. The knockoff flies off the shelves.
>
> J.K. Rowling sues for copyright infringement. Ann's defenses: 'But I wrote most of the words from scratch. Besides, this was fair use, because I copied only the portions necessary to tap into the Harry Potter fan base.'
>
> Obviously, the defenses would fail.[436]

Oracle's approach was reminiscent of an ultimately unsuccessful strategy from the first wave of API copyright litigation. Apple, IBM, and Lotus lawyers sought to compare creativity in the design and coding of computer software with conventional literary and dramatic works.[437]

However, the "software as creative expression" theme resonated with the Federal Circuit. The court's opinion repeatedly references the creativity of Java APIs.[438] The court pointed to the testimony of Joshua Bloch, the former Sun software engineer whom Google referred to as its "Java guru," who "conceded" that there can be "creativity and artistry even in a single method declaration."[439] The Federal Circuit offered its own literary metaphor, noting that "the opening of Charles Dickens' A TALE OF TWO CITIES is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work

---

436. *See* Opening Brief and Addendum of Plaintiff-Appellant at 12–13, Oracle Am., Inc. v. Google Inc., No. 2013-1021, 1022 (Fed. Cir. Feb. 11, 2013).

437. *See supra* note 90; Clapes, Lynch & Steinberg, *supra* note 90, at 1547.

438. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1352 (Fed. Cir. 2014) ("Although [the district court] acknowledged that the overall structure of Oracle's API packages is creative . . ."); *id.* at 1356 ("The testimony at trial revealed that designing the Java API packages was a creative process and that the Sun/Oracle developers had a vast range of options for the structure and organization."); *id.* ("In its copyrightability decision, the district court specifically found that the API packages are both creative and original, and Google concedes on appeal that the originality requirements are met."). *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp.2d 974, 976 (N.D. Cal. 2012) ("The overall name tree, of course, has creative elements . . . ."); *id.* at 999 ("Yes, it is creative. Yes, it is original."); *Oracle v. Google*, 750 F.3d at 1361, n.6 (noting that the Amicus Brief filed by Scott McNealy & Brian Sutphin "provide[d] a detailed example of the creative choices involved in designing a Java package"); *id.* at 1368 (observing that "Amici McNealy & Sutphin explain that 'a quick examination of other programming environments shows that creators of other development platforms provide the same functions with wholly different creative choices.'").

439. *Oracle v. Google*, 750 F.3d at 1339.

is unworthy of copyright protection because it can be broken into those shorter constituent components."[440]

The Federal Circuit reversed the district court's determination that the structure, sequence, and organization of the 37 Java APIs were not copyrightable and remanded the fair use issue for retrial with revised jury instructions.[441]

i. Copyrightability

In reviewing the district court's determination that the Java API packages at issue were not copyrightable, the Federal Circuit distinguished between copyrightability of the "declaring code" and copyrightability of the structure, sequence, and organization of the API packages.[442]

a. Declaring Code

The Federal Circuit ruled that the district court should not have considered the merger and *scènes à faire* doctrines in evaluating copyright subsistence because the Ninth Circuit treats these doctrines as affirmative defenses to infringement, not as limitations on copyrightability.[443] Hence, these doctrines were relevant only in determining what elements of the APIs should be filtered out in the infringement analysis.[444] Furthermore, the Federal Circuit held that the merger doctrine — which bars protection where an idea can only be expressed in one or a limited number of ways — properly focuses on the creative choices available to Sun when it created Java, not on the options available to Google when it copied Java APIs.[445] The Federal Circuit also held that the short phrases doctrine did not bar copyright protection for compilations of words and short phrases as reflected in the declaring code.[446] On these bases, the appellate court ruled copyright

440. *Id.*

441. *Id.*

442. *See id.*, at 1359–68.

443. *See id.* at 1358 (citing Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1082 (9th Cir. 2000)); Satava v. Lowry, 323 F.3d 805, 810 n.3 (9th Cir. 2003) ("The Ninth Circuit treats scènes à faire as a defense to infringement rather than as a barrier to copyrightability.").

444. *See Oracle v. Google*, 750 F.3d at 1359–62 (addressing the merger doctrine); *id.* at 1363–64 (addressing the *scènes à faire* doctrine, which Judge Alsup had rejected as a basis for holding the Java APIs to be unprotectable but that Google challenged on appeal).

445. *See id.* at 1360–61.

446. *See id.* at 1362–63. It should be noted that the district court's determination that the declaring code was uncopyrightable did not turn on the short phrases doctrine. Judge Alsup recognized that the selection and arrangement of short phrases could be protectable. *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 992 (N.D. Cal. 2012) (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 349 (1991) for the proposition that even thinly protected, factual compilations are protectable with respect to original "selection and arrangement"). His ultimate determination hinged on § 102(b) of the Copyright Act and interoperability. *See id.* at 997–1002.

law protected the 7,000 lines of declaring code. It did not directly confront the argument that the precise API declarations functioned as uncopyrightable "methods of operation," which more accurately characterizes Judge Alsup's essential holding. The Federal Circuit did, however, address the "method of operation" argument in its API SSO ruling.

b. SSO of the API Packages

The Federal Circuit focused its review of Judge Alsup's holding that the SSO of the Java APIs was uncopyrightable on the district court's reliance upon *Lotus v. Borland*,[447] the First Circuit case holding that the Lotus 1-2-3 menu command hierarchy was an unprotectable "method of operation." The appellate court distinguished *Lotus* on factual grounds, noting that the command labels at issue there, unlike the Java API declaring code, were "not creative" and were "essential" to operating the computer system.[448] Moreover, the Federal Circuit interpreted the Ninth Circuit's *Johnson Controls* to hold that the SSO of a computer program is eligible for copyright protection and hence was inconsistent with *Lotus*.[449] In so doing, the Federal Circuit resurrected the Third Circuit's flawed analytical framework: analyzing copyrightability of computer software based on whether the high level function(s) of the software could be implemented in multiple ways rather than viewing a particularized set of software functions as an unprotectable "method of operation."[450]

The Federal Circuit rejected the district court's invocation of interoperability as a basis for holding the SSO of the Java APIs to be uncopyrightable. Notwithstanding the language in *Sega* and *Sony* that the precise coding to achieve interoperability is not protectable under copyright law,[451] the appellate court distinguished these cases as "focused on fair use, not copyrightability."[452] The Federal Circuit repeated its earlier observation that "copyrightability is focused on the

447. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir.1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996).

448. *Oracle v. Google*, 750 F.3d at 1365.

449. *See id.* at 1365–66. The Federal Circuit's interpretation of *Johnson Controls* stretches its holding and overlooks important insights from later Ninth Circuit cases. *See infra* Section IV(A). In addition, the copyrightability of software SSO in some circumstances does not necessarily conflict with the exclusion of methods of operation.

450. *See id.* at 1366–67.

451. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1525 (9th Cir. 1992); Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 603 (9th Cir. 2000) ("There is no question that the Sony BIOS contains unprotected functional elements.").

452. *See* 750 F.3d at 1369 (observing that *Sega* and *Sony* never addressed whether the functional code had separable expressive elements). This assertion overlooks, however, that both courts ruled that the code necessary to interoperability was unprotectable and hence copying of the entirety of the software for purposes of reverse engineering the code to determine those interoperable features constituted fair use.

choices available to the plaintiff at the time [of] the computer program['s] [] creat[ion]," and not on the defendant's goal of achieving interoperability.[453] Therefore, Google's interoperability argument comes into play only as part of a fair use defense.

ii. Fair Use

The Federal Circuit was tempted to rule in Oracle's favor on the fair use issue.[454] The court observed that "[o]n many of [Oracle's] points,[455] Google does not debate Oracle's characterization of its conduct, nor could it on the record evidence."[456] Nonetheless, the Federal Circuit determined that remand was necessary because materials facts were in dispute, notably the transformativeness of the Android platform, Google's interoperability objectives, and the commercial impact of Android on Sun's/Oracle's mobile licensing activities and the potential market for a Java smartphone.[457] The Federal Circuit emphasized, however, that the district court should "revisit and revise its jury instructions on fair use consistent with [the Federal Circuit's] opinion."[458] The Federal Circuit's opinion did not, however, offer specific criticism of the district court's jury instructions.

4. Interlocutory Certiorari Petition

Google sought to challenge the Federal Circuit's reversal by filing a petition for a writ of certiorari with the U.S. Supreme Court.[459] Google's petition pressed the argument that the Java API declarations fall within the § 102(b) exclusion from copyright protection of methods of operation. Oracle responded that the case was not appropriate for interlocutory review on substantive and prudential grounds.[460] The Supreme Court nonetheless requested the views of the Solicitor General,[461] which produced one of the more surprising filings in the

453. *See id.* at 1370. The Federal Circuit follows the Third Circuit's dicta — "a defendant's desire 'to achieve total compatibility . . . is a commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged,'" *id.* (quoting *Apple Comput.*, 714 F.2d at 1253) — and not the rejection of that position in *Sega* and *Sony*. *See Sega*, 977 F.2d at 1525; *Sony v. Connectix*, 203 F.3d at 603.

454. *See* 750 F.3d at 1376.

455. *See id.* (noting that Oracle asserts that "Google knowingly and illicitly copied a creative work to further its own commercial purposes, and did so verbatim, and did so to the detriment of Oracle's market position").

456. *See id.*

457. *See id.* at 1377.

458. *See id.*

459. *See* Petition for a Writ of Certiorari, Google Inc. v. Oracle America, Inc., No. 14-410 (U.S. Oct. 6, 2014).

460. *See* Brief in Opp'n, Google Inc. v. Oracle Am., Inc., No. 14-410 (U.S. Dec. 8, 2014).

461. *See* Google Inc. v. Oracle Am., Inc., 135 S. Ct. 1021 (2015).

case.[462] The Solicitor General not only recommended against granting review on prudential grounds, but also sided with Oracle on substantive grounds.[463] The Supreme Court denied review.[464]

5. 2016 Fair Use Trial

The API copyright battle returned to Judge Alsup's court for a jury trial focused on applying "'the most troublesome [doctrine] in the whole law of copyright.'"[465] Google also planned to assert equitable estoppel and laches defenses.[466] Oracle expanded the scope of its complaint to account for new Android versions, Google's expansion into new product areas (clothing, television, automobiles, appliances, and media (Google Play)), and Android's dramatic market growth.[467]

Leading up to the trial, the parties squabbled over the fair use jury instructions.[468] After adjusting the draft instructions following input from the parties, one of the few, and most momentous, fair use jury trials in modern U.S. history commenced. Judge Alsup instructed the jury at the outset of the trial about the contours of the fair use doctrine, noting that the doctrine is an "equitable rule of reason" for which no generally accepted definition is possible.[469] He then read the statutory provision[470] and explained the four factors, boiling down the

462. *See* Brief for the United States as Amicus Curiae, Google Inc. v. Oracle Am., Inc., No. 14-410 (U.S. May 2015).

463. *See id.* at 11–17; *cf.* Dan Levine & Lawrence Hurley, *Google Versus Oracle Case Exposes Differences Within Obama Administration*, REUTERS (May 15, 2015), http://www.reuters.com/article/us-google-oracle-lawsuit-insight-idUSKBN0O017Z20150515 [https://perma.cc/GM3W-3AGZ].

464. Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015).

465. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1372 (Fed. Cir. 2014) (quoting Monge v. Maya Magazines, Inc., 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) (per curiam)).

466. *See* Google's Trial Brief at 11–12, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1706) (asserting that Sun's public statements and acts approving of Android's use of Java bar enforcement of its copyrights); ORDER RE WILLFULNESS AND BIFURCATION, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1321). The equitable defenses were bifurcated and hence did not arise during the fair use trial.

467. *See* PLAINTIFF ORACLE'S [PROPOSED] SUPPLEMENTAL COMPLAINT, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1288-1).

468. *See, e.g.*, Oracle Am., Inc., v. Google Inc., 118 U.S.P.Q.2d 1561 (N.D. Cal. 2016) (rejecting Google's request to include "as part of a broader work" within the jury instruction defining "transformative"); Oracle's Response To The Court's Request For Critique Re Instructions On Fair Use, Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA (N.D. Cal. filed Apr. 14, 2016) (ECF No. 1663).

469. *See* PENULTIMATE JURY INSTRUCTION ON FAIR USE, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1790).

470. *See* 17 U.S.C. § 107 (2012).

subtleties of the vast fair use jurisprudence into about a dozen treatise-like paragraphs.

The trial played out over eight grueling days of testimony ranging from the dramatic (embarrassing Emails) to the mind-numbing (experts and fact witnesses explaining API design, open source, GNU, GPL, virtual machines, and distinctions between declaring and implementing code).[471] The jurors were treated to creative and strained analogies (filing cabinets, breakfast menus featuring hamburgers, and Harry Potter novels), all manner of demonstrative exhibits, and a witness list featuring some of Silicon Valley's most celebrated tech billionaires. Economic experts opined about transformativeness (from an economic, as opposed to a legal, perspective) and network effects. Both sides made witnesses squirm. The connection of some lines of questioning to copyright law's fair use factors was often tenuous. For example, Oracle devoted much of its trial time to exposing Emails sent among Google engineers suggesting that they thought that the Java APIs were copyright-protected.[472]

In view of the large stakes — Oracle sought upwards of $10 billion in damages and injunctive relief[473] — both sides employed top-notch trial teams and spared little expense. The fair use trial felt like a roller coaster, with prognosticators divided on how the jury would come out.[474] Google bore the burden of proof on the fair use defense

---

471. The media and bloggers covered the trial extensively. I canvassed various sources, including: Joe Mullin's reporting for ARS TECHNICA; Sarah Jeong Storify (Twitter feed of Sarah Jeong, a contributing editor at Motherboard), https://storify.com/sarahjeong [https://perma.cc/P73V-DHDF]; FOSS PATENTS (blog published by Florian Mueller, a self-described "intellectual property activist"), http://www.fosspatents.com/; I also reviewed exhibits, such as pleadings, jury instructions, and slide decks.

472. *See id.*

473. *See* Joe Mullin, *Oracle Will Seek a Staggering $9.3 Billion in 2nd Trial Against Google*, ARS TECHNICA (Mar. 29, 2016) http://arstechnica.com/tech-policy/2016/03/oracle-will-seek-a-staggering-9-3-billion-in-2nd-trial-against-google/ [https://perma.cc/2X2Q-LGC3] (noting that Oracle sought the "biggest IP verdict ever").

474. *See Oracle vs. Google — The Merry-go-round*, RADIO FREE MOBILE (May 11, 2016), http://www.radiofreemobile.com/oracle-vs-google-the-merry-go-round/ [https://perma.cc/
LL75-YQN4] (predicting a settlement of less than $1 billion); Jeff Taylor, *Oracle v. Google: How to Create Beautiful Closing Argument Slides*, THE DROID LAWYER (May 26, 2016), http://thedroidlawyer.com/2016/05/oracle-v-google-how-to-create-beautiful-closing-argument-slides/ [https://perma.cc/JAP3-3R9C] (observing that Oracle's trial team's closing slides show that they "gathered the evidence they needed to prove their case"; but in a post-script, noting the irony that Oracle lost); Florian Mueller, Oracle v. Google *Copyright Retrial Won't Bring Clarification on Application Programming Interfaces (APIs)* (May 8, 2016), http://www.fosspatents.com/2016/05/oracle-v-google-copyright-retrial-wont.html [https://perma.cc/6MHU-RFGK] ("While I'm as convinced as ever that there is hardly a clearer case of UNfair use than this one . . . the trial is a tossup"; predicting a "55% or 60% chance for Google" before a jury, but that the appeals court "would be fairly likely to side with Oracle"); *cf.* Joel Rosenblatt, *Stakes Are High at Google vs. Oracle Copyright Trial #2*, INS. J. (May 10, 2016), http://www.insurancejournal.com/news/national/2016/05/10/407960.htm [https://perma.cc/2QWN-LURA] (quoting Professor Tyler Ochoa stating that it's a "fool's errand" to predict the outcome of the case with a new jury).

and hence, presented its case first. Judge Alsup limited each side to fifteen hours (nine hundred minutes) of testimony presentation time, including cross-examination. Each side was also afforded an hour for opening argument and ninety minutes for closing argument. Judge Alsup bifurcated the damages phase, which would be needed only if Google's use of Java APIs was not fair use.

i. Opening Arguments

Building upon the infringement ruling revived by the Federal Circuit, Oracle opened the second trial in rhythmic, Cochranesque[475] fashion: Google copied the heart of the Java platform so as to enter the mobile marketplace quickly and now seeks to use the "fair use excuse" to avoid the consequences.[476] Peter Bicks, Oracle's lead counsel, framed the battle in moralistic terms and epic proportions:[477]

- Internal e-mails show that Google took illegal "shortcuts" to create Android.
- "It took 10,000 lines to power this Apollo computer module, when lives were at stake. OVER ONE THOUSAND FEWER THAN WHAT GOOGLE COPIED."
- "Oracle was seeing money go out the door," while Google was earning billions on the Sun/Oracle investments in Java.
- "If [Java] code wasn't in their three billion phones, not one would work."

Drawing on its successful Federal Circuit strategy, Oracle characterized the crafting of the Java API code as highly creative, whereas Google's copying of Java APIs was slavish and not transformative. Bicks quoted liberally from internal Google Emails singing the praises of Java's APIs and expressing the need to obtain a license. He characterized the Android team's decision to forgo a license as underhanded — breaking the Write Once, Run Anywhere interoperability promise — and hence ineligible under the fair use doctrine's equitable standards.

Robert Van Nest, Google's lead counsel, emphasized Google's hard work and large investment in building a transformative

475. *See O. J. Simpson Murder Case*, WIKIPEDIA, https://en.wikipedia.org/wiki/O._J._Simpson_murder_case [https://perma.cc/88E6-3W8J] (noting defense attorney Johnny Cochran's quip "If [the glove] doesn't fit, you must acquit").

476. *See* Joe Mullin, *Google Took Our Property — and Our Opportunity, Oracle Tells Jury: "If that Code Wasn't in Their Three Billion Phones, Not One Would Work."* ARS TECHNICA (May 10, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-tells-jury-dont-buy-googles-fair-use-excuse/ [https://perma.cc/QN6S-9BDB].

477. *See id.*

smartphone platform.[478] He justified use of Java, in part, based on Sun's encouragement of the developer community to use Java and its APIs. He downplayed the expressive creativity of Java APIs by analogizing the API packages to the labels on a filing cabinet, carting a real filing cabinet into the courtroom to illustrate the point.[479] He emphasized that Sun's then-CEO Jonathan Schwartz publicly applauded Google's use of Java technologies in Android: Google had "strapped another set of rockets to the [Java] community's momentum — and to the vision defining opportunity across our (and other) planets,"[480] and that Oracle's CEO Larry Ellison welcomed Google's use of Java for its mobile platform.

Van Nest countered the allegation that Android caused Java's mobile platform to falter with an internal Oracle document pointing to its own internal problems, arguing that Oracle sued only after the Java mobile strategy failed to reap what Google had sown. He distinguished between the Java SE and ME platforms to highlight the transformativeness of Android's path-breaking approach. Java ME was a "feature phone" platform,[481] whereas Android brought the functionality of robust web browsing, apps, and a host of other functionalities such as cameras and games (e.g., Angry Birds) to mobile devices. Van Nest displayed a graphic showing that Java code represented a very small percentage, less than one-tenth of a percent, of the Android code base. Furthermore, Google developed its own virtual machine for Android devices.

Van Nest also sought to sow the seed of a new fair use factor or sub-factor: compliance with industry norms surrounding APIs. Although not one of the four express statutory fair use factors, in Google's view API declaring code was fair game so long it was implemented independently (i.e., clean room), especially where the platform developer had welcomed platform adopters. He concluded his opening by arguing that:

> Android is precisely the kind of thing that fair use was intended to encourage. It's a leap forward to a new platform in a new market. It has allowed inno-

478. *See* Joe Mullin, *Google to Jury: Android Was Built with Our Engineers' Hard Work*: *"Android Is Precisely the Kind of Thing that Fair Use Was Intended to Encourage,"* ARS TECHNICA (May 10, 2016), http://arstechnica.com/tech-policy/2016/05/google-to-jury-android-was-built-with-our-engineers-hard-work/ [https://perma.cc/XZS8-MMKJ].

479. *See* Sarah Jeong, *In a $9 Billion Trial, Google's Secret Weapon Is a Filing Cabinet*, VICE: MOTHERBOARD (May 11, 2016), http://motherboard.vice.com/read/googles-lawyers-tried-to-explain-apis-to-a-jury-using-a-physical-filing-cabinet [https://perma.cc/4G23-CH8T]. This analogy echoed earlier API copyright cases, notably *Apple v. Microsoft* (desktop icons of the graphical user interface) and *Lotus v. Borland* (spreadsheet command labels).

480. *See Congratulations Google*, *supra* note 180.

481. *See supra* text accompanying note 261.

> vation by lots and lots of other people — developers and wireless carriers. It's become a whole community, because Google made it open and free. Now Mr. Ellison wants to shut it down and put it in his pocket. That is not fair, not right, and not what copyright was intended to allow.[482]

ii. Google's Case in Chief

Google began its testimony with Eric Schmidt, Google's Chairman and Sun's former Chief Technology Officer at the time that Java was developed.[483] Schmidt discussed Sun's encouragement of Java adoption as well as his understanding that Google was free to use the Java APIs without a license. On cross-examination, Oracle sought to undermine Schmidt's rosy characterization of the Sun-Google relationship and highlighted Google's reputation for pushing to "the creepy line" in business tactics.[484]

Google then called Jonathan Schwartz, who enthusiastically explained that Java had always been free and open.[485] Schwartz testified that Sun promoted Java's use to build a community of developers throughout the world and counter Microsoft's power in the desktop operating system marketplace. Schwartz further explained that the Java APIs were also free for others to use and independently implement.

Schwartz bizarrely analogized APIs to hamburgers on a breakfast menu:[486] different restaurants offer the same item, but they have their

---

482. *Id.*

483. *See* Joe Mullin, *On the Stand, Google's Eric Schmidt Says Sun Had No Problems with Android*, ARS TECHNICA (May 10, 2016), http://arstechnica.com/tech-policy/2016/05/oracles-lawyer-grills-googles-eric-schmidt-on-the-nature-of-apis/ [https://perma.cc/32KM-F8RY]; Sarah Jeong, *Oracle v. Google — Day 1* (May 10, 2016), https://storify.com/sarahjeong/oracle-v-google-day-1 [https://perma.cc/45QC-PPR6]; Sarah Jeong, *Oracle v. Google — Day 2* (May 10, 2016), https://storify.com/sarahjeong/oracle-v-google-day-2-5748d3691f6a1e6260160f34 [https://perma.cc/RN4V-ETRN].

484. *See* Derek Thompson, *Google's CEO: 'The Laws Are Written by Lobbyists,'* THE ATLANTIC (Oct. 1, 2010), http://www.theatlantic.com/technology/archive/2010/10/googles-ceo-the-laws-are-written-by-lobbyists/63908/ [https://perma.cc/F9YL-F5JL].

485. *See* Joe Mullin, *Sun's Jonathan Schwartz at trial: Java was free, Android had no licensing problem*, ARS TECHNICA (May 11, 2016), http://arstechnica.com/tech-policy/2016/05/suns-jonathan-schwartz-at-trial-java-was-free-android-had-no-licensing-problem/ [https://perma.cc/Y6R8-APG4]; Sarah Jeong, *Oracle v. Google — Day 1* (May 10, 2016), https://storify.com/sarahjeong/oracle-v-google-day-1 [https://perma.cc/45QC-PPR6]; Sarah Jeong, *Oracle v. Google — Day 2*, (May 10, 2016), https://storify.com/sarahjeong/oracle-v-google-day-2-5748d3691f6a1e6260160f34 [https://perma.cc/RN4V-ETRN].

486. *Cf.* Sarah Jeong, *In Oracle v. Google, a Nerd Subculture Is on Trial*, MOTHERBOARD (May 12, 2016), http://motherboard.vice.com/read/in-google-v-oracle-the-nerds-are-getting-owned [https://perma.cc/ZB6Y-YPBD] (noting Judge Alsup's statement that "[t]he thing about the breakfast menu makes no sense," and commenting that "[n]o one

own implementation — i.e., their own way of preparing the quintessential American sandwich. According to Schwartz, Sun's strategy was to offer open APIs and compete on implementations. He noted that this sometimes undermined Sun's control, such as when the free software community developed the GNU Classpath project, a free implementation of the standard class library for the Java programming language, without a license.[487] Schwartz testified that he was "annoyed, but it was completely consistent with our practices. When you say APIs are open, there are competitive implementations."[488] Schwartz also discussed the Apache Harmony platform, supported by a coalition including IBM, Oracle, and Google, which modestly forked the Java platform without a license.[489] Schwartz acknowledged that Sun's only control was through trademark protection: "It wasn't going to call itself Java, so there was nothing we could do." He noted, however, that all of the projects promoted use of the Java language, which enhanced Sun's reputation and leadership.

Schwartz also testified about Sun's failure to introduce its own mobile phone product and his disappointment that Sun and Google did not reach a licensing arrangement that could have enhanced Sun's reputation in the marketplace. He denied, however, that Android contributed to Sun's failure to develop a Java-based smartphone.

On cross-examination, Oracle challenged Schwartz's objectivity and business acumen. Schwartz acknowledged that Oracle had not offered him a senior management position following its acquisition of Sun and that he was not aware that Sun had entered into a specification license with Apache regarding the Harmony platform. Bicks brought out internal Sun Emails showing great frustration with Google's unwillingness to partner on a mobile platform and concern that Android would undermine the interoperability of the Java platform. Schwartz explained that he was trying to put a positive public face ("make lemonade with lemons"[490]) on a difficult business circumstance. He acknowledged his consternation with Google: "They

---

bothered to challenge Schwartz's apparent belief that hamburgers are commonly featured on breakfast menus").

487. *See Apache Harmony*, WIKIPEDIA, https://en.wikipedia.org/wiki/Apache_Harmony [https://perma.cc/U6A3-WNWF].

488. *See* Joe Mullin, *Sun's Jonathan Schwartz At Trial: Java Was Free, Android Had No Licensing Problem*, ARS TECHNICA (May 11, 2016), https://arstechnica.com/tech-policy/2016/05/suns-jonathan-schwartz-at-trial-java-was-free-android-had-no-licensing-problem/ [https://perma.cc/VPK6-HGRN].

489. *Id.*

490. *See* Dan Farber, *Java Creator James Gosling: 'Google Totally Slimed Sun,'* CNET (Apr. 30, 2012) (quoting Gosling stating that "[w]e were all really disturbed, even Jonathan [Schwartz]: he just decided to put on a happy face and tried to turn lemons into lemonade."), http://www.cnet.com/news/java-creator-james-gosling-google-totally-slimed-sun/ (last visited Jan. 27, 2018).

take Java without attribution or contribution. That is why I love Scroogle."

Google then called Andy Rubin, leader of the Android project, to the witness stand.[491] He explained Google's vision of creating an open smartphone platform where Google would profit not from the sale of devices or software but from promoting its web services and advertising platform. He denied that Java was necessary for Android's success, but acknowledged that it accelerated its entry into the marketplace. Rubin explained his understanding that Android could not use the Java trademarks without a license, but admitted that his team could independently implement the Java APIs.

Annette Hurst, Oracle's co-lead counsel, put Rubin through a relentless, aggressive cross-examination lasting more than four hours aimed at establishing that Google took shortcuts and knowingly copied Java APIs in developing the Android platform. Rubin acknowledged that he stood to earn $60 million by getting the Android smartphone to market by specified milestones. Much of the cross-examination explored Rubin's Emails, first seeking to work out a Java platform license that would enable the Android team to pursue its open source model, and then, after negotiations reached an impasse, strategizing about independently implementing the Java APIs.

By the end of a full day of cross-examination, Oracle had burned through much of its allotted time and had not yet begun its case in chief. Judge Alsup warned Oracle that he did not plan to grant additional time.

Google presented video deposition excerpts in which Larry Ellison denied saying he found Android's use of Java flattering and did not recall saying that he was excited about more Java-based products coming from his friends at Google.[492] Google then played a video from the Java One Conference in which Ellison made both statements. The deposition excerpts further showed Oracle did not pursue a mobile smartphone device and that Java had continued to grow since Android's release. Google then introduced deposition testimony from an IBM executive explaining that IBM uses the unlicensed Apache Harmony implementation of Java SE.

---

491. *See* Joe Mullin, *Copyright and consequences: Google's Andy Rubin Defends Android to Jury*, ARS TECHNICA (May 12, 2016), http://arstechnica.com/tech-policy/2016/05/copyright-and-consequences-googles-andy-rubin-defends-android-to-jury/ [https://perma.cc/4JC9-YBUV]; Sarah Jeong, *Oracle v. Google — Day 2* (May 12, 2016), https://storify.com/sarahjeong/oracle-v-google-day-2-5748d3691f6a1e6260160f34 [https://perma.cc/RN4V-ETRN]; Sarah Jeong, *Oracle v. Google — Day 3* (May 12, 2016), https://storify.com/sarahjeong/oracle-v-google-day-3 [https://perma.cc/T5FB-DUEZ].

492. *See* Joe Mullin, *Top Programmer describes Android's Nuts and Bolts in Oracle v. Google*, ARS TECHNICA (May 14, 2016), http://arstechnica.com/tech-policy/2016/05/top-programmer-describes-androids-nuts-and-bolts-in-oracle-v-google [https://perma.cc/QAE6-FXZU]; Sarah Jeong, *Oracle v. Google — Day 4* (May 13, 2016), https://storify.com/sarahjeong/oracle-v-google-day-4 [https://perma.cc/X52E-ZAKR].

Google called Joshua Bloch, the former Sun employee who became Google's "Java guru." Bloch played a significant role in developing Java APIs and authored EFFECTIVE JAVA,[493] a book about writing Java code. Bloch explained the goals of API design (to make them concise and difficult to misuse) and Sun's desire to make them widely available. He discussed differences in writing APIs for mobile, as opposed to desktop, environments. Drawing on Bloch's writings, Oracle focused its cross-examination on the creativity involved in designing APIs. He acknowledged that writing good APIs is difficult.

Google played video deposition excerpts of Donald Smith, a designated Oracle representative,[494] in which Smith stated the Java programming language and the Java APIs were defined together under the same specification and hence were inseparable. He further testified that there were more than ten million Java developers and Oracle's Java division was growing and profitable. In a later segment of the deposition, Smith walked back his earlier testimony that the Java language and APIs were inseparable.

Google next called Simon Phipps, who was previously Sun's Chief Open Source Officer and was also President of the Open Source Initiative until 2015.[495] Phipps testified that Sun had not taken actions to stop other projects that used Java APIs such as GNU Classpath and Apache Harmony.

Google then called Daniel Bornstein, a key member of the Android development team, to discuss APIs and the Android team's approach to using Java declarations and APIs. Bornstein considered Java declarations "A-OK to use." He explained that Google used a lot of open source software, including Apache Harmony "core libraries," to build Android. He noted that no other product offered the functionality, such as running multiple applications simultaneously on a smartphone, that Google sought to develop. On cross-examination, Hurst questioned Bornstein about Google's efforts to purge java-related terms from the Android code. Bornstein made light of the suggestion that this indicated that Sun owned the APIs. On re-direct, Bornstein explained that he was not a lawyer and had called for scrubbing the "J-word" (Java) from Android code to avoid trademark concerns.[496]

---

493. *See* JOSHUA BLOCH, EFFECTIVE JAVA (2001); JOSHUA BLOCH, EFFECTIVE JAVA (2d ed. 2008)

494. *See* FED. R. CIV. P. 30(b)(6).

495. *See Open Source Initiative*, WIKIPEDIA, https://en.wikipedia.org/wiki/Open_Source_Initiative [https://perma.cc/J88L-5Y5W].

496. *See* Joe Mullin, *At Trial, Top Android Coder Explains Oracle's Questions on "Scrubbed" Source Code*, ARS TECHNICA (May 16, 2016), http://arstechnica.com/tech-policy/2016/05/at-trial-top-android-coder-explains-oracles-questions-on-scrubbed-source-code/ [https://perma.cc/HRR3-G4SL]; Sarah Jeong, *Oracle v. Google — Day* 5 (May 16,

Google completed its direct fair use case with Professor Owen Astrachan, Professor of the Practice of Computer Science at Duke University.[497] Professor Astrachan provided clear and measured testimony about API design, the distinction between declaring and implementing code, and the importance of consistent functional labels in programming.[498] He explained that Android is not fully compatible with Java SE because the SE platform is designed for desktop or laptop computers whereas Android is designed for mobile devices. Google designed Android to make use of 37 well-known Java APIs. Since Java is the most widely used computer program in the world, "[d]evelopers would expect that if you're going to be using the Java programming language, you'd have access to a rich suite of APIs, to write whatever program you're going to write." Professor Astrachan illustrated that the Java API labels (declarations) are functional and descriptive — discussing java.net (network classes); java.io (input/out), java.sql (accessing and processing data stored in a data source, usually a relational database), java.security (classes and interfaces for the security framework), and java.util (various collections of functions, including date and time and internationalization).[499] He explained the GNU Classpath implementation of Java APIs and the clean room process. He further noted that Sun had reimplemented the Linux APIs in its Solaris platform.

On cross-examination, Hurst pressed Professor Astrachan on the creativity involved in designing APIs. While agreeing that designing a good API is difficult, Astrachan observed that the difficulty was "not exactly" the same as that encountered by painters or musicians.[500] He acknowledged that the Java language did not require the selection of the particular 37 Java APIs that Google incorporated in Android, but that it was necessary to meet developer expectations.

2016), https://storify.com/sarahjeong/oracle-v-google-day-2 [https://perma.cc/T5MU-GZEX].

497. *See* Owen Astrachan, DUKE U., https://users.cs.duke.edu/~ola/ [https://perma.cc/K83A-NWRP]; *Owen Astrachan*, WIKIPEDIA, https://en.wikipedia.org/wiki/Owen_Astrachan [https://perma.cc/9PW6-4Y9H].

498. *See* Joe Mullin, *Google Puts Its Expert on the Stand to Combat Oracle, Wraps Up Its Case*, ARS TECHNICA (May 16, 2016), http://arstechnica.com/tech-policy/2016/05/google-puts-its-expert-on-the-stand-to-combat-oracle-wraps-up-its-case/ [https://perma.cc/XE4V-VLS6]; Sarah Jeong, *Oracle v. Google — Day 5* (May 16, 2016), https://storify.com/
sarahjeong/oracle-v-google-day-2 [https://perma.cc/T5MU-GZEX].

499. *See* Appendix A.

500. *See* Joe Mullin, *Google puts its expert on the stand to combat Oracle, wraps up its case*, ARS TECHNICA (May 16, 2016), http://arstechnica.com/tech-policy/2016/05/google-puts-its-expert-on-the-stand-to-combat-oracle-wraps-up-its-case/ [https://perma.cc/XE4V-VLS6].

iii. Oracle's Case in Chief

By the time that Google completed its case, Oracle had used much of its allotted time cross-examining Google's witnesses. Oracle opened its case in chief with Oracle co-CEO, Safra Catz.[501] She explained that Oracle acquired Sun to ensure the stability and reliability of Java, on which many of Oracle's software products were built. She testified that "Java was the single most important asset Oracle ever acquired."[502] She denied that Oracle sought to pursue a copyright infringement lawsuit against Google. Catz explained the importance of intellectual property protection to support Oracle's $5.5 billion annual investment in research and development. She discussed how Android's forking of Java code had undermined Oracle's licensing strategy. On cross-examination, Catz acknowledged that Sun had licensed "significant elements" of Java technology as open source, which could reduce the ability to appropriate revenue from users.

Oracle next called two other company executives.[503] Edward Screven, Oracle's chief corporate architect, reinforced Catz's testimony regarding Oracle's motivation for acquiring Sun. He also explained that the Apache Harmony license required that the Apache license meet the Java Technology Compatibility Kit ("TCK") test suite and hence was not equivalent to Android's use.[504] He testified that Android was the only unlicensed use of Apache Harmony.

Oracle next called Mark Reinhold, Oracle's chief architect for Java SE, in what may have been the most significant testimony in the case. Reinhold noted that the APIs for the Java ME (Micro Edition, for feature phones) contain the same structure, sequence, and organization as those of Java SE (for desktop computers). Drawing on Oracle's Federal Circuit strategy, Reinhold testified that "the Java API Package is like a book series" as the Harry Potter series flashed on the courtroom presentation screen. He developed the following syllogism:

---

501. *See* Joe Mullin, *Oracle CEO Safra Catz: "We Did Not Buy Sun to File this Lawsuit,*" ARS TECHNICA (May 16, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-ceo-safra-catz-we-did-not-buy-sun-to-file-this-lawsuit/ [https://perma.cc/7HP3-SQ8Y]; Joe Mullin, *Oracle CEO: Google's Android Broke Java in Two*, ARS TECHNICA (May 17, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-ceo-googles-android-broke-java-in-two/ [https://perma.cc/V2WC-CLYK]; Sarah Jeong, *Oracle v. Google — Day 6* (May 17, 2016), https://storify.com/sarahjeong/oracle-v-google-day-7 [https://perma.cc/8BAQ-3AJX].

502. *See* Mullin, *supra* note 501.

503. *See* Joe Mullin, *Oracle Java Architect Conscripts* Harry Potter *in Making the Case Against Google*, ARS TECHNICA (May 17, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-java-architect-conscripts-harry-potter-in-making-the-case-against-google/ [https://perma.cc/T2FJ-LVVU]; Sarah Jeong, *Oracle v. Google — Day 6* (May 17, 2016), https://storify.com/sarahjeong/oracle-v-google-day-7 [https://perma.cc/8BAQ-3AJX].

504. *See Technology Compatibility Kit*, WIKIPEDIA, https://en.wikipedia.org/wiki/Technology_Compatibility_Kit [https://perma.cc/ELH6-KC3D].

Package = Book
Class = Chapter
Method = Paragraph

Reinhold explained that Google's copying of the Java API declaring code is:

> [L]ike using the titles of the books, the headings of each chapter, and the title sentences of each paragraph as well as the connections between the characters. Three books later, there are all these deep connections. It's intensely creative. Like writing a book, you have to keep a lot of stuff in your head, and the end result is rich and complex. A lot of it is about figuring out what structures you want.[505]

Reinhold dismissed Van Nest's analogy of Java APIs to labels on a filing cabinet as "laughably simplistic."

On cross-examination, Google pressed Reinhold on the incompatibility across Java various platforms, getting him to acknowledge that Java ME would not pass the Java SE compatibility test. Reinhold also acknowledged that Java SE did not scale down for smaller devices, implicitly acknowledging that Android provided an innovative new platform.

Oracle then called Douglas Schmidt, Professor of Computer Science at Vanderbilt University, as an expert witness.[506] Professor Schmidt presented a visual software map illustrating the interconnectedness of the APIs at issue. He testified that Google used the 37 Java APIs in the same way that Sun designed them for the Java platform. He corroborated Reinhold's testimony that the APIs at issue were "creative" and "substantial." He presented test results showing that Android failed if any of the Java APIs or the declaring code were removed. Schmidt put into context Google's claim that the Java declaratory code represented less than one-tenth of one percent of Android's fifteen million lines of code by illustrating that more than sixty percent of the Android code was copied from third-parties. Furthermore, of the twenty-three percent of the Android code that Google wrote, nine percent were blank or comment lines. On cross-examination, Professor Schmidt acknowledged that he was not familiar with the meaning of "free and open" source software when he began preparing his testimony.

505. *See* Mullin, *supra* note 503.
506. Professor Schmidt is not related to Google's executive chairman.

Oracle completed its fair use case with testimony about the economic impacts of Android's release.[507] Neil Civjan, Sun's head of global sales, testified that there were 2.6 billion Java-enabled mobile phones at the peak (85% of the global marketplace). That number fell precipitously after the introduction of Android's phones and its freely licensed operating system. Civjan noted that Java licensees did not see why they should license the Java ME platform when they could get Android, which was essentially Java and Linux, for free. He characterized the effect on Sun's licensing business as "devastating."

Alan Brenner, Sun's Senior Vice President of client systems from 1997 until 2007, corroborated Civjan's testimony and testified that Sun had persuaded a Korean research institute to take a Java license rather than use the GNU Classpath project. Brenner rebutted Jonathan Schwartz's testimony that Sun accepted other implementations of the Java platform. On cross-examination, Brenner acknowledged that Java licensing revenue was in decline before Android launched.

Oracle called Stefano Mazzocchi, a Google engineer who was one of the original Apache Harmony developers, to rebut Google's argument that Sun acceded to others' use of the Java APIs.[508] Apache Harmony obtained a license, subject to restrictions, on its use of the Java platform. Following the announcement of Oracle's acquisition of Sun in 2009, Mazzocchi emailed members of the Apache listserv with his concerns about Java's future: "What is Oracle going to do about Android's ripping off some of (now) their IP and getting away with it?"[509] In an earlier Email, Mazzocchi expressed the view that copyright protected the Java APIs:

> But what I was missing is the fact that the copyright on the API is real and hard to ignore.
> Simply by implementing a class with the same signature of another, in another namespace and simply by looking at available javadocs could be considered copyright infringement, even if the implementation is clean room.

---

507. *See* Joe Mullin, *Sun's Head of Java Sales: Android Was "Devastating*," ARS TECHNICA (May 18, 2016), https://arstechnica.com/tech-policy/2016/05/suns-head-of-java-sales-android-was-devastating/ [https://perma.cc/3EE8-P2AU].

508. *See* Joe Mullin, *Apache E-mails, Shown in Court, Say Android "Ripped Off" Oracle IP*, ARS TECHNICA (May 18, 2016), http://arstechnica.com/tech-policy/2016/05/apache-e-mails-shown-in-court-say-android-ripped-off-oracle-ip./ [https://perma.cc/5YMC-U82V].

509. *See* Email from Stefano Mazzocchi to members@apache.org (Apr. 20, 2009), Trial Ex. 9201, Oracle Am., Inc. v. Google Inc., (No. C 10-03561 WHA), http://arstechnica.com/wp-content/uploads/2016/05/9201.pdf [https://perma.cc/V73L-DBSL]

> So, we are, in fact, infringing on the spec lead copyright if we distribute something that has not passed the TCK and *we know that*.[510]

On cross-examination, Mazzocchi acknowledged that he was neither a lawyer nor an expert on copyright law.

Oracle concluded its case by calling Professor Adam Jaffe, an economics expert, to explain network effects and his conclusion that Android was not transformative from an economic perspective.[511] Professor Jaffe testified that Android "very likely would not have been successful" had Google not copied the 37 Java APIs. He further opined that Java was "poised to enjoy continued success" in the mobile marketplace. But because of network effects, the market quickly tipped toward the Android platform and Sun was unable to recover. Professor Jaffe contended that Java ME supported smartphones, but was unable to gain traction in Android's wake. On cross-examination, Van Nest used the Java ME-based SavaJe phone (see Figure 6), which lacked a QWERTY keyboard or touch screen, to illustrate the stark differences between the Java ME platform and the Android platform. Professor Jaffe acknowledged that SavaJe was a failure.

---

510. *See* Email from Sam Ruby to members@apache.org (Apr. 17, 2008), Trial Ex. 5046, Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA)), http://arstechnica.com/wp-content/uploads/2016/05/5046_REDACTED.pdf [https://perma.cc/BX79-5BE2] (including Email from Mazzocchi in Email thread).

511. *See* Joe Mullin, *Oracle Economist: Android Stole Java's "Window of Opportunity,"* ARS TECHNICA (May 18, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-economist-android-stole-javas-window-of-opportunity/ [https://perma.cc/J8VM-6QMB]; Sarah Jeong, *Oracle v. Google — Day 7* (May 18, 2016), https://storify.com/sarahjeong/oracle-v-google-day-7-573d5aff5cb000d21eb9311d [https://perma.cc/99MG-QRTQ].

Figure 6
Jave ME-based SavaJe Phone

Figure 6. Java ME-Based SavaJe Phone.

iv. Google's Rebuttal

Google first called Larry Page, Google's co-founder and CEO of Alphabet, Google's parent corporation, who testified that Google never believed that it needed a license for the Java APIs because they were "free and open."[512] On cross-examination, Page acknowledged that unauthorized use of Google's intellectual property could harm the company. He did not believe, however, that API declarations constituted computer code. He reiterated that he considered the Java APIs to be free and open. He acknowledged, however, that he was not a lawyer and did not "know the vagaries of licensing."

Google then called Dr. Greg Leonard, an economics expert, to respond to Professor Jaffe's testimony.[513] Dr. Leonard concluded that

512. *See* Joe Mullin, *CEO Larry Page Defends Google on the Stand: "Declaring Code is Not Code"*, ARS TECHNICA (May 19 2016), http://arstechnica.com/tech-policy/2016/05/ceo-larry-page-defends-google-on-the-stand-declaring-code-is-not-code/ [https://perma.cc/85QW-TMS2].

513. *See* Joe Mullin, *Oracle v. Google Draws to a Close, Jury Sent Home Until Next Week*, ARS TECHNICA (May 19, 2016), http://arstechnica.com/tech-policy/2016/05/oracle-v-google-draws-to-a-close-jury-sent-home-until-next-week/ [https://perma.cc/7AZE-RSBS].

Android did not have any impact on licensing of Java ME because feature phones were not substitutes for smartphones. He further opined that use of the thirty-seven APIs was not "central to Android's success"; in his view, C++ could have done comparably well.

Google completed the testimony phase of the trial by recalling Professor Owen Astrachan, its programming expert. Professor Astrachan was not at all surprised that Android failed to operate with the Java declaring code removed. He then summarized Google's approach to designing Android:

(1) Google selected 37 (not all) packages from Java SE, and used those method declarations;

(2) wrote implementing code for those declarations;

(3) added other libraries specific to smartphones, like GPS, camera, etc.;

(4) brought in third-party libraries for stuff like web browsers and graphics;

(5) made the Dalvik Virtual Machine; and

(6) built whole thing on top of Linux.

In his expert opinion, this effort produced an innovative, open source mobile platform.

v. Closing Arguments

The final day of the trial began with Judge Alsup reading the jury twenty-one pages of instructions: general instructions regarding evidence, witnesses, credibility, and burden of proof (seven pages); established facts regarding the copyrighted works at issue (three pages); the meaning of fair use under copyright law (eight pages); and jury deliberation procedures (three pages).[514] The fair use instructions mirrored the instructions set forth at the outset of the trial. Judge Alsup allotted each side ninety minutes for closing argument.

Van Nest began by emphasizing that this case was very important not only for Google, but for innovation and technology in general.[515]

514. *See* Notice of Final Charge to the Jury (Phase One) and Special Verdict Form, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1928).

515. *See* Joe Mullin, *Google's Closing Argument: Android Was Built from Scratch, the Fair Way*, ARS TECHNICA (May 23, 2016), http://arstechnica.com/tech-policy/2016/05/googles-closing-argument-android-was-built-from-scratch-the-fair-way/ [https://perma.cc/U2XD-CE3C]; Sarah Jeong, *Oracle v. Google - Closing Arguments* (May 23, 2016), https://storify.com/sarahjeong/oracle-v-google-closing-arguments [https://perma.cc/YK4U-CWTN].

He then noted that the clear, consistent, and largely uncontested testimony of Eric Schmidt, Jonathan Schwartz, Andy Rubin, and Joshua Bloch established that Sun made Java free and open to use and encouraged widespread use. Google took up that invitation and developed a pathbreaking mobile platform. Google independently implemented the selected Java APIs, resulting in a software platform that used less than one half of one percent of Java code. Van Nest laid blame for this litigation squarely on Oracle Chairman Larry Ellison, who Van Nest asserted brought this case after he had tried to use Java to build his own smartphone and failed.

Van Nest then launched into his core copyright defense: "Android is exactly the kind of thing the fair use doctrine was supposed to protect." Van Nest emphasized Android's transformative purpose: it is not a substitute for Java SE or Java ME, but rather is an innovative smartphone platform. Furthermore, Sun invited others to use Java. Van Nest characterized the Java APIs as functional, reminding the jury of the filing cabinet labels. He recalled Schwartz's hamburger implementation metaphor. Van Nest noted that Android had not interfered with the market for Java SE and that the Java language remained the most popular coding language in the world. Van Nest concluded with an industry custom argument — every witness acknowledged that re-implementing APIs was common in the software industry.

In response, Bicks returned to simple, moralistic themes: "You don't take people's property without permission and use it for your own benefit"; you don't take "shortcuts" at other people's expense; the "fair use excuse."[516] Bicks methodically built Oracle's closing around the "mountain of evidence," principally Emails that Google engineers never thought would see the light of day. He deployed a professionally-crafted storyboard to illustrate Oracle's fair use analysis.[517]

In constructing the argument against fair use, Bicks emphasized the clear commerciality of Google's use of the Java APIs and the extensive copying — 11,500 lines of Java code. He emphasized the Har-

---

516. *See* Joe Mullin, *Oracle slams Google to jury*: *"You don't take people's property"*, ARS TECHNICA (May 23, 2016), /http://arstechnica.com/tech-policy/2016/05/oracle-slams-google-to-jury-you-don't-take-peoples-property [https://perma.cc/BN7H-2SAJ]; Sarah Jeong, *Oracle v. Google — Closing Arguments* (May 23, 2016), https://storify.com/sarahjeong/oracle-v-google-closing-arguments [https://perma.cc/YK4U-CWTN].

517. *See* Joe Mullin, *How Oracle Made Its Case Against Google*, *in Pictures: Armed with Google's Own E-mails, Oracle Said "Fair Use" Was Nowhere to be Found*, ARS TECHNICA (May 25, 2016), http://arstechnica.com/tech-policy/2016/05/how-oracle-made-its-case-against-google-in-pictures/ [https://perma.cc/AA5Y-7URS]; Jeff Taylor, *Oracle v. Google: How to Create Beautiful Closing Argument Slides*, THE DROID LAWYER (May 26, 2016), http://thedroidlawyer.com/2016/05/oracle-v-google-how-to-create-beautiful-closing-argument-slides/ [https://perma.cc/CC77-KCMV].

ry Potter metaphor to illustrate the rich, "creative," integrated design of the Java APIs, as reflected in Figure 7.

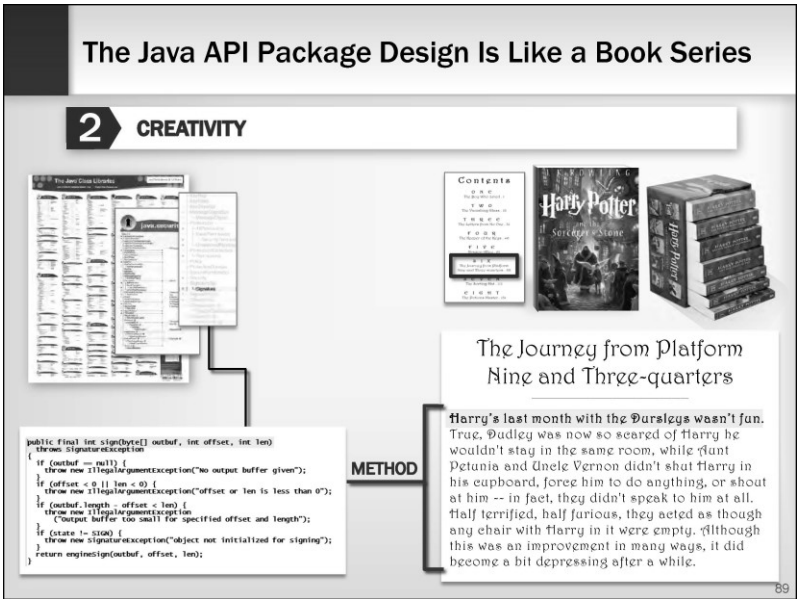

Figure 7. Oracle's Closing Argument: HARRY POTTER Metaphor

Bicks mocked Google's opportunistic use of Schwartz's congratulatory blog post after Android was announced by showing a two-faced silhouette juxtaposing Sun's public face with his cynical, internal face (referring to Google as "Scroogle"), as Figures 8 and 9 show. He characterized Google as a bully and Oracle as a courageous fighter standing up to Google's arrogance.

Figure 8. Oracle Closing Argument: Two Faces of Jonathan Schwartz, "Public versus Private"

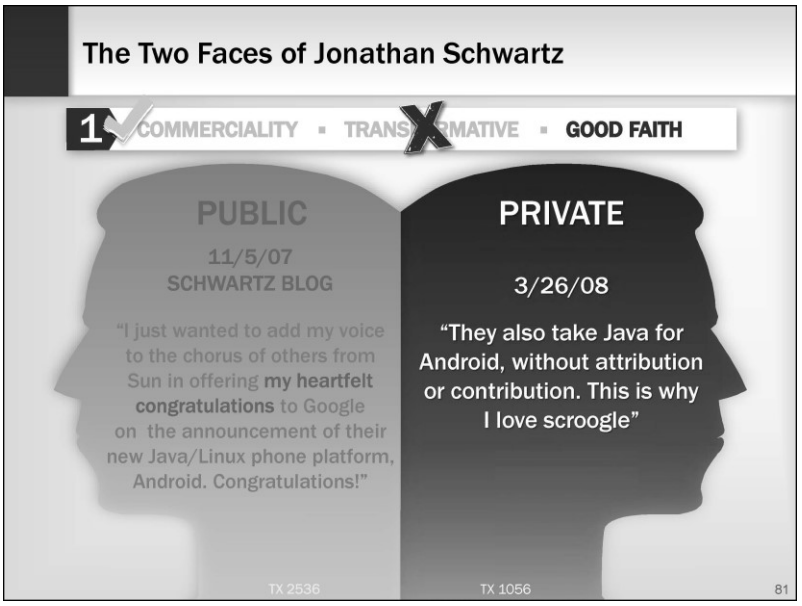

Figure 9. Oracle Closing Argument: Two Faces of Jonathan Schwartz, "Scroogle"

Bicks rebutted Google's suggestion that Java APIs were free and open, pointing out that Judge Alsup had instructed the jury that only the Java language was open and free. The court specifically instructed the jury that the Java language required only 170 of the 11,500 lines of API declaring code that Google copied.[518]

Bicks countered the suggestion that Android makes transformative use of the Java APIs by emphasizing that Android used the Java API packages to effectuate the same purposes as the Java platform (e.g., java.security for security).[519] Bicks further noted that Java ME provided a full-stack solution for smartphones, as reflected in its use in SavaJe and other functioning, although not commercially successful, smartphones.[520] In Oracle's view, Java ME's decline was due to its free availability in a marketplace heavily influenced by network economics (i.e., tipping point).[521]

Bicks concluded with the property theft theme: "Imagine, somebody takes your property and is then competing against you — for free."[522] Even if Sun and Oracle stumbled in building a smartphone platform, that did not justify Google taking their property: "Maybe you have some land and build a barn on it, and it doesn't stand up that well. Somebody doesn't get to come onto your property, and say, 'You weren't good at building a barn, so I'm going to build a barn here.' The evidence isn't that Oracle failed. Android took over the market."[523]

Bicks sought to leave the jury with a bitter taste by emphasizing that fair use presupposes good faith and fair dealing. In Oracle's view, Google played by its own self-serving rules.[524]

In Google's rebuttal, Van Nest countered that Sun gave away the Java APIs with the Java language to promote the language.[525] He ridiculed Oracle's reliance upon Emails among engineers about the law and conflation of trademark (scrubbing the J-word) and copyright issues. Van Nest acknowledged that all companies have internal debates, but that Google properly concluded that the API declarations were not copyrightable and were available to be re-implemented in a transformative platform.[526] Van Nest deflected the stealing and theft

518. Judge's Instructions/Charge to the Jury at *2209, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1973).

519. Plaintiff Oracle America, Inc.'s Closing Statement and Defendant's Rebuttal, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1973).

520. *Id.*

521. *Id.*

522. *Id.*

523. *Id.*

524. *Id.*

525. Defendant Google Inc.'s Closing Statement, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1973).

526. *Id.*

arguments as beside the point of a fair use trial.[527] He mocked the "shortcut" argument by pointing out that it took Google five years to bring Android to market. Van Nest countered Oracle's moralistic stealing theme with the argument that widespread and long-standing industry norms supported independent implementation of APIs.[528]

Google closed the trial by suggesting that transformativeness provides the sensible middle ground between stolen and free. "You don't have to choose between commercial and transformative . . . . [b]ecause the whole purpose of fair use is to promote innovation."[529]

vi. Jury Verdict

Following three days of deliberation, the jury found that Google had "shown by a preponderance of the evidence that its use in Android of the declaring lines of code and their structure, sequence, and organization from Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0 constitutes a 'fair use' under the Copyright Act."[530] The verdict form did not ask the jury to make subsidiary factual findings.[531] With fair use decided in Google's favor, there was no need for a further damages phase. Judge Alsup thanked the jury for their hard work and discharged the ten jurors.[532] The jurors departed without comment, leaving the public and the appellate court without a clear understanding of how the fair use balance was struck.

6. The Road Ahead

As the *Oracle v. Google* litigation illustrated, a jury verdict does not necessarily resolve a dispute, especially where the cost of appeal is relatively low in comparison with the stakes involved and the parties perceive no advantage to settlement.[533] As Google completed its case in chief, Oracle filed a motion requesting that Judge Alsup render judgment as a matter law ("JMOL") in its favor.

---

527. *Id.*
528. *Id.*
529. *Id.*
530. *See* Special Verdict Form, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA); Joe Mullin, *Google Beats Oracle — Android Makes "Fair Use" of Java APIs*, ARS TECHNICA (May 26, 2016), http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/ [https://perma.cc/J325-V5MS].
531. *See* Special Verdict Form, *supra* note 530 .
532. *See* Mullin, *supra* note 530.
533. *See supra* notes 366–75.

Judge Alsup rejected Oracle's JMOL motion.[534] He explained that he erred on Oracle's side in allowing an instruction on the propriety of the defendant's conduct[535] notwithstanding both the Federal Circuit's failure to call attention to this consideration in its remand decision and the Supreme Court's decision in *Campbell v. Acuff-Rose Music, Inc.*, which downplays or jettisons this consideration.[536] He further explained that based on the evidence presented, the jury could well have determined that it was fair use to maintain the same structure of 37 Java API packages in the Android re-implemented packages so as to avoid the confusion that would ensue from scrambling the various functions: "avoiding cross-system babel promoted the progress of science and useful arts — or so our jury could reasonably have found."[537]

Judge Alsup rejected Oracle's arguments that Android's use of the Java APIs should have been deemed "entirely commercial" and non-transformative, and that the Java APIs should have been considered "highly creative" because of the myriad ways in which the functions could have been implemented. With respect to the fourth fair use factor — the impact on the potential market for the Java platform — Judge Alsup ruled that the jury "could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers" and that the copying had little effect on licensing of Java ME beyond "the tailspin already predicted within Sun."[538] The court concluded its ruling by highlighting the contradiction between Oracle's pretrial instruction arguments — focusing on characterizing the fair use test as an equitable rule of reason affording juries broad discretion based on the contextual facts of the case — and its JMOL motion urging that the court override the jury's balancing of the fact-specific factors:

---

534. *See* Order Denying Rule 50 Mots. at 1, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

535. *See* Notice of Final Charge to the Jury (Phase One) and Special Verdict Form at 14, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

536. *See* 510 U.S. 569, 585 n.18 (1994) ("Even if good faith were central to fair use, 2 Live Crew's actions do not necessarily suggest that they believed their version was not fair use; the offer [to license the plaintiff's work] may simply have been made in a good-faith effort to avoid this litigation. If the use is otherwise fair, then no permission need be sought or granted."); 2 PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 12.2.2, at 12:44.5–12:45 (3d ed. 2016).

537. Order Denying Rule 50 Mots., *supra* note 534, at 8–10. Judge Alsup further explained that inter-system consistency "differs from the interoperability point criticized by the Federal Circuit. 750 F.3d at 1371. The immediate point of cross-system consistency focuses on avoiding confusion in usage between the two systems, both of which are Java-based, not on one program written for one system being operable on the other, the point addressed by the Federal Circuit." Order Denying Rule 50 Mots., *supra* note 534, at 10 n.6.

538. *See* Order Denying Rule 50 Mots., *supra* note 534, at 17.

> In applying an 'equitable rule of reason,' our jury could reasonably have given weight to the fact that cross-system confusion would have resulted had Google scrambled the SSO and specifications. Java programmers and science and the useful arts were better served by a common set of command-type statements, just as all typists are better served by a common QWERTY keyboard.[539]

That decision did not, however, end even the trial court phase of the litigation. Oracle filed a new JMOL motion in early July that largely critiqued Judge Alsup's rejection of its first JMOL motion.[540] More significantly, Oracle filed a motion requesting a new trial based on Google's alleged failure to disclose its plan to install Android Marshmallow on desktop and laptop computers.[541] In its reply to Google's opposition,[542] Oracle contended that the withheld evidence "directly refutes Google's argument to the jury that 'Android is not a substitute [because] Java SE is on personal computers; Android is on smartphones.'"[543]

Judge Alsup rejected these motions but left open the option for Oracle to file a new copyright infringement complaint based upon Google's implementations of Android in devices other than smartphones and tablets in a separate proceeding and trial.[544] Oracle has appealed the trial court's verdict and post-trial determinations.

Oracle has reason for optimism about winning a Federal Circuit appeal.[545] Pursuant to the Federal Circuit's Internal Operating Proce-

---

539. *See* Order Denying Rule 50 Mots., *supra* note 534, at 18.

540. *See* ORACLE'S RULE 50(b) MOT. FOR JUDGMENT AS A MATTER OF LAW, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1993).

541. *See* ORACLE'S RULE 59 MOT. FOR A NEW TRIAL, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 1995-5).

542. *See* GOOGLE INC.'S OPP'N TO ORACLE'S RULE 59 MOT. FOR A NEW TRIAL, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 2012).

543. *See* ORACLE'S REPLY IN SUPP. OF ITS RULE 59 MOT. FOR A NEW TRIAL, at 1, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) (ECF No. 2018-2) (citing Trial Tr. at 2124:6-7 (Google Closing Argument)).

544. *See* Order Denying Renewed Mot. For Judgment As A Matter Of Law And Mot. For A New Trial, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

545. *See* Florian Mueller, Oracle v. Google: *Jury Finds in Favor of "Fair Use," As No Reasonable, Properly-instructed Jury Could Have*, FOSS PATENTS (May 26, 2016), http://www.fosspatents.com/2016/05/oracle-v-google-jury-finds-in-favor-of.html [https://perma.cc/3N3S-HWV5] (contending that Judge Alsup's instructions set the fair use bar far too low). *But see* Jonathan Band, *Sanity Prevails Again, Part II: The District Court Leaves the* Oracle v. Google *Fair Use Verdict in Place*, DISRUPTIVE COMPETITION PROJECT (Jun.

dures, the same panel that reversed Judge Alsup's copyrightability ruling and set forth guiding principles for the fair use trial heard the appeal of the fair use trial.[546] Oracle preserved various objections to Judge Alsup's jury instructions.[547] And since Judge Alsup denied Oracle's new trial motion, Oracle has further grounds for appealing the fair use verdict. Moreover, the appellate panel has already indicated that there was much force to Oracle's position and that many of the facts relevant to the fair use balance were not in dispute.[548]

Google also has reason for optimism. First, it won the jury trial after Judge Alsup modified the jury instructions in light of the parties' concerns. Second, even if Google were to lose at the Federal Circuit level a second time, it could petition the Supreme Court to review the Federal Circuit's API copyrightability ruling.[549]

Assuming that the parties don't reach a settlement, which has proven especially difficult, the Federal Circuit will review the fair use trial and post-trial rulings. Should Google prevail, Oracle would likely take a shot at Supreme Court review. Alternatively, the Federal Circuit could remand for another fair use trial or resolve the ultimate fair use question in Oracle's favor, thereby setting up a Google writ of certiorari petition raising both API copyrightability and fair use questions. Under the most optimistic scenario, the case will continue for several years. Furthermore, all new uses of Android could attract new claims of copyright infringement.[550]

10, 2016), http://www.project-disco.org/intellectual-property/061016-sanity-prevails-again-part-
ii-the-district-court-leaves-the-oracle-v-google-fair-use-verdict-in-place/#.V7sha_krJph [https://perma.cc/DVW7-AJKC] (contending that "given how the district court meticulously found evidence in the record supporting the reasonableness of the jury's fair use finding, it is hard to imagine that the Federal Circuit will reverse it").

546. *See* U.S. Court of Appeals for the Federal Circuit Internal Operating Procedures, Rule #3 (Merits Panels — Distribution of Briefs, Records, and Files (Nov. 14, 2008) ("When an appeal is docketed in a case that was previously remanded by this court . . . the clerk's office attempts to assign the appeal to the previous panel, to a panel including at least two members of the previous panel (if one of those members was the authoring judge), or to a panel that contains the authoring judge, if such a panel is otherwise constituted and available on a subsequent argument calendar."), http://www.cafc.uscourts.gov/sites/default/files
/IOPs122006.pdf [https://perma.cc/ZT93-L52X].

547. *See* Order Denying Rule 50 Mots., Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA).

548. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1376 (Fed. Cir. 2014).

549. *See* Google's Trial Brief at 8 n.12, Oracle Am., Inc. v. Google Inc., 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA) ("Google does not waive and hereby expressly preserves its position that the SSO/declarations are not protected by copyright law. *See, e.g.*, Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032 (9th Cir. 2015).").

550. *See, e.g.*, Florian Mueller, *Three Angles to Look at Google's Pixel Phone: Design Patents, Antitrust, Copyright*, FOSS PATENTS (Oct. 28, 2016), http://www.fosspatents.com/2016/10/three-angles-to-look-at-googles-pixel.html [https://perma.cc/W7NL-LKCD] (not-

C. The Current Murky State of API Copyright Protection

The *Oracle v. Google* fair use jury trial ranks among the most significant computer software intellectual property trials and copyright fair use trials in U.S. history. Yet, it provided little clarity to what is an especially murky area of intellectual property law. Even though Google has prevailed thus far, the jury's fair use decision has little precedential significance. Even if the higher courts leave this verdict intact, other technology companies will be left to roll the dice if they incorporate unlicensed re-implemented APIs in their platform specification. Furthermore, Google faces exposure for new versions of Android that implement Java APIs in new products. The jury's verdict in *Oracle v. Google* does not insulate them from the risk of being sued for copyright infringement. The only secure safe harbors are to develop an independent platform or license the pre-existing APIs.

The Federal Circuit's decision rejecting Judge Alsup's API copyrightability ruling is the most significant recent federal appellate decision to confront the copyrightability of APIs. Furthermore, given the proliferation of software patents, a company with a widely used set of APIs could very likely pursue both patent and copyright causes of action in the same litigation,[551] thereby bringing the Federal Circuit's exclusive jurisdiction over patent cases into play, even where patent issues are not appealed.

Thus, notwithstanding six years of litigation and two jury trials, the *Oracle v. Google* litigation has contributed to, rather than quelled, confusion surrounding API copyright protection. As courts have noted, fair use is "'the most troublesome [doctrine] in the whole law of copyright.'"[552] Legal advisors will need to inform their clients that there is no clear safe harbor for re-implementing APIs short of a license. Other trial teams will face the same troublesome doctrines in the context of another set of complex facts.

Furthermore, by resolving the fair use question with a simple jury verdict form, the *Oracle v. Google* litigation sheds little light on the reasoning behind the jury's decision. There were no formal factual findings. Therefore, the decision contributes little to our understand-

---

ing that Oracle could assert a new copyright complaint against Google's new Pixel smartphone, which implements Java APIs on Android Nougat).

551. *See* Scott Graham, Cisco v. Arista *IP Battle Starts to Look a Lot Like* Oracle v. Google, THE RECORDER (Aug. 26, 2016), http://www.therecorder.com/id=1202766017854/Cisco-v-Arista-IP-Battle-Starts-to-Look-a-Lot-Like-Oracle-v-Google?slreturn=20160905152607 (last visited Jan. 27, 2018).

552. *See Oracle*, 750 F.3d at 1372 (quoting Monge v. Maya Magazines, Inc., 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) (per curiam)); *see also* PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 12.1 (3d ed. 2005) ("No copyright doctrine is less determinate than fair use."); David Nimmer, *"Fairest of Them All" and Other Fairy Tales of Fair Use*, 66 LAW & CONTEMP. PROBS. 263, 263 (2003).

ing of the fair use factors — transformativeness, commerciality, nature of the copyrighted work — or how they are balanced in the context of new platforms building on and augmenting prior API packages. All we know is that Google's particular re-implementation for particular products was fair use. But as Judge Alsup's resolution of the new trial reveals, further development of the Android platform could well provide the basis for a new copyright infringement action.

Such uncertainty can be especially problematic for technology companies. The viability and value of a platform depends critically upon its ability to leverage consumers' and programmers' familiarity with APIs. Hence, the design of a new platform requires planning and coordination. Yet the current status of API copyright jurisprudence hinges liability for copyright infringement on "'the most troublesome [doctrine] in the whole law of copyright.'"[553]

The unusual jurisdictional posture of the *Oracle v. Google* case and other API disputes that arguably implicate patent protection further complicates the API copyright puzzle. When Congress established the Court of Appeals for the Federal Circuit in 1982,[554] it did not provide a procedure for reviewing Federal Circuit interpretations of regional circuit law short of Supreme Court review. The Federal Circuit is the only en banc process available to litigants. It would be more appropriate, however, to present such issues to the regional circuit, especially in cases such as *Oracle v. Google* in which patents play no role in the appellate proceeding.[555] Such a review would be analogous to certification of a state law question to the highest state court. Yet Congress has not authorized such review. As a result, the Federal Circuit's exclusive jurisdiction over federal patent law cases produces a dual body of regional circuit law. The extent to which such decisions bind the regional circuit is unclear since there are no structural means to harmonize divergent appellate interpretations short of Supreme Court review.

The *Oracle v. Google* case illustrates the "forking"[556] of Ninth Circuit copyright jurisprudence. Whereas Judge Alsup placed principal reliance on the Ninth Circuit's *Sega* decision, which expressly rejected the *Whelan* framework, the Federal Circuit emphasized its *Nintendo v. Atari Games* decision, which predates *Sega* and builds on an inchoate foundation of the Ninth Circuit's *Johnson Controls* decision. Technology companies are left without a clear line of authority

553. *See Oracle*, 750 F.3d at 1372.

554. *See* Federal Courts Improvement Act of 1982, Pub. L. No. 97-164, 96 Stat. 25 (1982).

555. *See* Peter S. Menell, *API Copyrightability Bleak House: Unraveling and Repairing the* Oracle v. Google *Jurisdictional Mess*, 31 BERKELEY TECH. L.J. 1515 (2016).

556. *See supra* note 16; *see also* Appendix A (defining forking).

or a procedure for resolving such differences unless the Supreme Court intervenes.

The following Part critically analyzes the *Oracle v. Google* litigation and constructs a coherent framework for applying copyright law to APIs.

IV. THE LAW AND ECONOMICS OF API COPYRIGHT PROTECTION

Congress's decision to bring computer software within the scope of copyright protection was never intended to hinder technological innovation. The legislative history of the 1976 Copyright Act as well as the CONTU REPORT made clear that copyright law's limiting principles were an essential part of Congress's calculus in affording computer software copyright protection. In keeping with the long-standing common law traditions of copyright law, courts would play a critical role in applying and adapting copyright law's limiting doctrines to take account of technological change.

The early history of copyright protection for computer software technology illustrates the courts' role in fitting copyright protection for computer software within the contours of the larger intellectual property system. It is not surprising that courts struggled with the early cases. Few judges were familiar with computer technology and the software marketplace was developing rapidly. By the early 1990s, scholarship, experts, and advocates provided judges with a richer understanding of how copyright protection for software technology fit within the larger intellectual property system. The proper balance reflected the interplay of technological innovation and interoperability as well as the distinct and complementary roles of copyright and patent protection.

The *Altai* case provided a robust framework for limiting copyright protection to the non-functional elements of computer software. The *Sega* case, reinforced by the interoperability provisions of the DMCA, established that interoperable features of computer technology were fair game for subsequent software developers so long as they implemented the functional specifications in independently written code. By the mid-1990s, a coherent body of software copyright law had emerged.

The network and other functional features of computer software were not eligible for copyright protection even as the thousands of lines of implementing code garnered copyright protection against piracy. This balance operationalized the wisdom of the *Baker v. Selden* case and the useful article separability doctrine in the software copy-

right domain. Litigation subsided, and the software industry moved forward.[557]

The *Oracle v. Google* litigation revived flawed and widely rejected arguments from the first wave of API copyright litigation. The Federal Circuit's decision finding that compilations of functions in API packages as well as the structure, sequence, and organization of APIs are protectable so long as there are multiple ways of achieving the high-level purposes of the software returns us to the *Apple v. Franklin* and *Whelan* era. This type of regime effectively protects particular machines under copyright law so long as there are multiple methods to implement those machines' general functions. Such broad copyright protection intrudes upon the functional realm reserved for utility patent protection.

Under the Federal Circuit's *Oracle* ruling, companies that control widely adopted platforms can leverage copyright protection to control the investments of programmers and users of their technology. They can stand in the way of subsequent innovators that seek to effectuate a leap to a new functional paradigm. With three decades of experience in software platform evolution, we have a sounder basis for assessing the proper balance between promoting network externalities and encouraging platform innovation.

This Section reexamines the role of copyright protection for computer software in the current and foreseeable digital age. Section A critically analyzes the *Oracle v. Google* decisions and explains that copyright law's fundamental exclusion of protection for functional features dictates that the labeling conventions and packaging of functions within interface specifications generally fall outside of the scope of copyright protection even as implementing code garners thin copyright protection. Section B explains that this interpretation of copyright law serves the larger goals of intellectual property law and competition policy.

A. Legal Analysis

This Section begins by reviewing the foundational principles guiding copyright protection for computer software. It then assesses

---

557. *See* Baker v. Selden, 101 U.S. 99 (1879). Just as copyright protection for computer software became coherent, patent protection for computer software and business methods emerged as a major problem for the software industry. *See* Peter S. Menell, *Forty Years of Wondering in the Wilderness and No Closer to the Promised Land:* Bilski*'s Superficial Textualism and the Missed Opportunity to Return Patent Law to its Technology Mooring*, 63 STAN. L. REV. 1289 (2011); JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK (2009); ADAM B. JAFFE & JOSH LERNER, INNOVATION AND ITS DISCONTENTS: HOW OUR BROKEN PATENT SYSTEM IS ENDANGERING INNOVATION AND PROGRESS, AND WHAT TO DO ABOUT IT (2004).

the Federal Circuit's *Oracle* decision. It concludes with a comprehensive framework for adjudicating software copyright cases.

1. Overarching Principles

The intellectual property system channels innovative, creative, and source-identifying works among three distinct modes of protection: utility patent law protects technological works; copyright law protects expressive works;[558] and trademark law protects source-identifying symbols. The requirements for eligibility, scope, duration, and remedies for each of the modes of protection vary significantly based on the differing underlying purposes and legislative design of patent, copyright, and trademark protection.

To a first approximation, technological and creative works have generally fallen into different modes of protection. Machines, technical processes, and chemical compositions are eligible for utility patent protection (or trade secret protection if maintained as secrets).[559] Literary, pictorial, graphic, sculptural, and musical works are protected through copyright law. Trade symbols are protected as trademarks, although a graphic symbol might also garner copyright protection.

The challenge computer software and other useful articles pose is that they can fall into two or more of the intellectual property modes. Patent-eligible machines can be characterized as sculptural works or source-identifying trade dress. Software code for running a machine can be characterized as literary text. The Supreme Court cogently resolved this overlap when it recognized that only utility patent law, the most restrictive of the intellectual property regimes, protects a work's functional features. Otherwise, inventors could effectively extend their statutory exclusive rights beyond the limited times that Congress intended for technological innovations and applied scientific discoveries. As the Supreme Court explained in *Baker v. Selden*,

> The copyright of the book, if not pirated from other works, would be valid without regard to the novelty, or want of novelty, of its subject-matter. The novelty of the art or thing described or explained has nothing to do with the validity of the copyright. To give to the author of the book an exclusive property in the art described therein, when no examination of its

558. Design patent law can also be used to protect the ornamental (non-functional) aspects of useful articles. *See* 17 U.S.C. § 171 (2012). It co-exists and overlaps with copyright protection. *See* Mazer v. Stein, 347 U.S. 201, 217 (1954).

559. Trade secret law protects information that derives value from not being generally known and is subject to reasonable efforts to maintain secrecy. *See* Uniform Trade Secrets Act § 1(4).

> novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright.[560]

A book describing a technological method can be the subject of a copyright without impinging on the public's use of the method taught and illustrated in text and pictures. As the Supreme Court summarized, "[t]here is a clear distinction between the book, as such, and the art which it is intended to illustrate. The mere statement of the proposition is so evident, that it requires hardly any argument to support it."[561] The Court further explained,

> A treatise on the composition and use of medicines, be they old or new; on the construction and use of ploughs, or watches, or churns; or on the mixture and application of colors for painting or dyeing; or on the mode of drawing lines to produce the effect of perspective, — would be the subject of copyright; but no one would contend that the copyright of the treatise would give the exclusive right to the art or manufacture described therein.[562]

This foundational channeling principle frames the intellectual property system. Without this principle, the potential overlaps among patent, copyright, and trademark protection would topple the edifice. Any patent-eligible method, machine, article of manufacture, or chemical composition can be described in a book. Any machine or article of manufacture can serve as an indicator of source. The long duration and low threshold requirements of copyright and trademark protection would displace patent's primacy in protecting technological advance or functional features. Inventors could use copyright or trademark protection to easily secure rights in technological advances for substantially longer duration than utility patent protection. Thus, the courts have barred copyright or trademark protection for methods, machines, and functional elements of sculptural works.[563] This same rationale preempts state laws aimed at directly protecting technology.[564]

---

560. 101 U.S. 99, 102 (1879).

561. *Id.*

562. *Id.*

563. *See id.*; TrafFix Devices, Inc. v. Mktg. Displays, Inc., 532 U.S. 23 (2001); Inwood Labs, Inc. v. Ives Labs, Inc., 456 U.S. 844, 863 (1982) (White, J., concurring in result) (explaining that where an item in general circulation is unprotected by patent, "[r]eproduction of a functional attribute is legitimate competitive activity.").

564. *See* Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 159–64 (1989); Sears, Roebuck & Co. v. Stiffel Co., 376 U.S. 225, 233 (1964) (barring state law from offer-

Congress expressly codified these doctrines in the 1976 Copyright Act. Section 102(b) provides that "[i]n 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."[565] The Copyright Act excludes "mechanical or utilitarian aspects" of useful articles from the definition of "pictorial, graphic, and sculptural works."[566] The statute provides that "the design of a useful article . . . shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article."[567]

The legislative history of the 1976 Copyright Act states that Congress's purpose in enacting § 102(b) was "to restate, in the context of the new single Federal system of copyright, that the basic dichotomy between expression and idea remains unchanged."[568] These limitations developed through judicial decisions, such as *Baker v. Selden*, and have produced a body of common law doctrines, such as merger, *scènes à faire*, and fair use. Congress intended to perpetuate judicial evolution of these doctrines as a means of adapting copyright law to technological change.[569]

Regarding copyright protection for computer software, the legislative history comments that:

> Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the program-

---

ing "the equivalent of a patent monopoly" in the functional aspects of a product which had been placed in public commerce absent the protection of a valid patent); Compco Corp. v. Day-Brite Lighting, Inc., 376 U.S. 234 (1964). State trade secret protection does not, in the Supreme Court's view, conflict with the federal patent regime. *See* Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470 (1974). Rather, trade secret protection focuses on misappropriation of secret information. It does not stand in the way of scientific discovery or technological innovation. The public may freely use knowledge that is not protected by patents, including information gleaned through reverse engineering of publicly available protections. *Id.* at 490.

565. 17 U.S.C. § 102(b).

566. 17 U.S.C. § 101 (definition of "pictorial, graphic, and sculptural works").

567. *Id.*

568. *See* H.R. REP. NO. 94-1476, at 57 (1976).

569. *See id.* at 66 ("The bill endorses the purpose and general scope of the judicial doctrine of fair use, but there is no disposition to freeze the doctrine in the statute, especially during a period of rapid technological change. Beyond a very broad statutory explanation of what fair use is and some of the criteria applicable to it, the courts must be free to adapt the doctrine to particular situations on a case-by-case basis."); *see generally*, Peter S. Menell, *The Mixed Heritage of Federal Intellectual Property Law and Ramifications for Statutory Interpretation*, *in* INTELLECTUAL PROPERTY AND THE COMMON LAW 70 (Shyamkrishna Balganesh ed., 2013).

> mer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.[570]

The Computer Software Copyright Act of 1980,[571] implementing CONTU's recommendations, affirmed CONTU's emphasis on the importance of applying the judicially-developed idea-expression doctrine to ensure that copyright protection did not interfere with technological progress in computer programming.[572]

The 1976 Copyright Act, as amended by the 1980 software amendments, put courts in the critical role of adapting copyright law's traditional, judicially-developed standards to the rapidly developing medium of computer software. Over the course of the next two decades, the courts rose to the challenge. After some initial missteps, which threatened to provide undue legal protection to the first entity to develop computer software for a particular purpose (such as managing a dental laboratory's records),[573] courts came to apply the idea-expression doctrine and other critical limiting doctrines with fuller appreciation of the purposes underlying copyright protection and its interplay with patent protection. The Ninth Circuit was especially forward-thinking in ensuring a proper balance.[574]

2. Critique of the Federal Circuit Copyrightability Decision

The Federal Circuit's *Oracle v. Google* decision purports to apply Ninth Circuit jurisprudence to its review of Judge Alsup's decision holding that the compilation of functions and the structure, sequence, and organization of the Java APIs were not copyrightable. This Section shows that the Federal Circuit (1) misinterpreted § 102(b) of the Copyright Act, (2) misconstrued Ninth Circuit software copyright jurisprudence, (3) conflated technological innovation and expressive or artistic "creativity," (4) applied an overly rigid approach to copyright law's limiting doctrines, and (5) treated API design as variable expression rather than unique function.

---

570. *See* H.R. REP. NO. 94-1476, at 57 (1976).

571. Pub. L. No. 96-517, 94 Stat. 3007, 3028 (codified at 17 U.S.C. §§ 101, 117).

572. *See supra* Section II(B)(1).

573. *See supra*, text accompanying note 90 (discussing Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.).

574. *See supra* notes 122–42.

i. Misinterpretation of the Copyright Act

The Federal Circuit's opinion takes a broad view of the scope of copyright protection for computer software.[575] While recognizing the § 102(b) limitations, the court did not view those constraints as applicable to copyrightability.[576] Rather, the court saw § 102(b) as only applicable at the infringement and defenses stages of analysis.

The Federal Circuit misread the clear language of the Copyright Act as well as the legislative history. It also misapprehends the larger legislative intent and purpose regarding copyright protection for useful articles and other functional subject matter.

a. Misreading Section 102

Section 102 of the Copyright Act addresses "Subject matter of copyright: In general." Section 102(a) sets forth a broad list of categories, such as literary works, musical works, and pictorial, graphic, and sculptural works, in which copyright protection subsists.[577] Section 102(b) sets forth limitations on copyrightable subject matter: "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."

Google argued that the particular compilation of functions in Java API packages were uncopyrightable "method[s] of operation." The Federal Circuit rejected the proposition that § 102(b) can be invoked in this way, quoting a comment in the legislative history of the 1976 Act stating that § 102(b) "in no way enlarges or contracts the scope of copyright protection," but merely "restates . . . that the basic dichotomy between expression and idea remains unchanged."[578] The Federal Circuit then turned to a Tenth Circuit case, *Mitel, Inc. v. Iqtel, Inc.*,[579] for the proposition that "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."[580] From there, the Federal Circuit, following *Mitel*, concluded that § 102(b) only

575. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1356 (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 345, 358 (1991)) (noting that originality "means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity" and that "the originality requirement is not particularly stringent").

576. *Id.* at 1354 (finding that the "district court failed to distinguish between the threshold question of what is copyrightable — which presents a low bar — and the scope of conduct that constitutes infringing activity.").

577. 17 U.S.C. § 102(a).

578. *Oracle*, 750 F.3d at 1356 (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 356 (1991) (quoting H.R. REP. No. 94-1476)).

579. 124 F.3d 1366 (10th Cir. 1997).

580. *Oracle*, 750 F.3d at 1356–57 (quoting *Mitel*, 124 F.3d at 1372).

comes into play as part of the abstraction-filtration-comparison analysis.[581]

The Federal Circuit overrode the plain text of the 1976 Act — "In no case does copyright protection for an original work of authorship extend to any . . . method of operation . . . , regardless of the form in which it is described, explained, illustrated, or embodied in such work"[582] — based on the comment in the legislative history that "Section 102(b) in no way enlarges or contracts the scope of copyright protection." As the Federal Circuit recognizes, however, Congress intended § 102(b) to codify the idea-expression dichotomy.[583] Under that doctrine, methods of operation — such as the accounting method in *Baker v. Selden* — were categorically excluded from copyright eligibility. The Supreme Court did not inquire into whether there were other methods that achieved the same purpose (accounting). Rather, the Court excluded any claim to a method of accounting even as it ruled that Selden's accounting book describing the method was copyrightable.[584]

Reinforcing this understanding of the idea-expression dichotomy, the CONTU REPORT declared that "one is always free to make a machine perform any conceivable process (in the absence of a patent)" so long as they don't "take another's program."[585] Following this principle, it is difficult to understand why Google would not be entitled to make a mobile device ("a machine") perform the same functions as a Java API package (a "conceivable process") with clean-roomed computer code (not "another's program"). Each Java API package constituted a particular subsystem within a larger particular computing environment. Extrapolating one step further, it is difficult to understand why Google would not be entitled to select a set of Java API packages and implement them with original code to create a new machine.

Congress directly addressed the interplay of copyright protection for computer software and the idea-expression dichotomy in the fol-

---

581. *See id.* at 1357.

582. 17 U.S.C. § 102(b).

583. *See Oracle*, 750 F.3d at 1355.

584. The Federal Circuit twisted *Baker v. Selden* in its atextual reading of § 102:

> The [Supreme] Court [in *Baker v. Selden*] indicated that, if it is necessary to use the forms Selden included in his books to make use of the accounting system, that use would not amount to copyright infringement. *See* [*Baker v. Selden*, 101 U.S. at 104] (noting that the public has the right to use the account-books and that, 'in using the art, the ruled lines and headings of accounts must necessarily be used as incident to it').

Oracle v. Google, 750 F.3d at 1355. A faithful reading of *Baker v. Selden* recognizes that the Court held that the accounting method was uncopyrightable, not merely not infringed. That is the essence of the idea-expression dichotomy.

585. *See* CONTU REPORT at 20.

lowing passage from the House Report: "Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law."[586] This language, unlike the general statement about that "Section 102(b) in no way enlarges or contracts the scope of copyright protection," captures the essence of API design. The implementing code is the protectable computer program. The declaring code constitutes "the actual processes or methods embodied in the program [which] are not within the scope of the copyright law."[587] This construction of § 102(b) is faithful to the text and specific legislative history of the Copyright Act.

b. Legislative Intent and Purpose

The Copyright Act's provisions relating to useful articles and general legislative history reinforce § 102(b)'s role as a threshold doctrine, not merely an infringement or fair use consideration.

The definition of "pictorial, graphic, and sculptural works" states that "the design of a useful article . . . shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article."[588] Congress plainly viewed the separability test as a threshold issue. The legislative history explains:

> [T]he Committee is seeking to draw as clear a line as possible between copyrightable works of applied art and uncopyrighted works of industrial design. . . . [A]lthough the shape of an industrial product may be aesthetically satisfying and valuable, the Committee's intention is not to offer it copyright protection under the bill. Unless the shape of an automobile, airplane, ladies' dress, food processor, television set, or any other industrial product contains some element that, physically or conceptually, can be identified as separable from the utilitarian aspects of that article, the design would not be copyrighted under the bill. The test of separability and independence from 'the utilitarian aspects of the article' does not depend upon the nature of the design — that is, even

586. *See* H.R. REP. NO. 94-1476, at 56–57 (1976).
587. *Id.*
588. 17 U.S.C. § 101 (definition of "pictorial, graphic, and sculptural works").

> if the appearance of an article is determined by esthetic (as opposed to functional) considerations, only elements, if any, which can be identified separately from the useful article as such are copyrightable.[589]

The functional characteristics of a work come into play in the infringement and fair use analyses if a useful article is physically or conceptually separable — i.e., if the entirety of the work is not categorically excluded at the copyrightability stage of analysis.

The 1976 Copyright Act legislative history also excludes typefaces from copyright protection out of concern with interfering with utilitarian functions:

> The Committee has considered, but chosen to defer, the possibility of protecting the design of typefaces. A "typeface" can be defined as a set of letters, numbers, or other symbolic characters, whose forms are related by repeating design elements consistently applied in a notational system and are intended to be embodied in articles whose intrinsic utilitarian function is for use in composing text or other cognizable combinations of characters. The Committee does not regard the design of typeface, as thus defined, to be a copyrightable "pictorial, graphic, or sculptural work" within the meaning of this bill and the application of the dividing line in section 101.[590]

Congress intended a similar threshold exclusion for functional elements of architectural works that are not separable from the artistic features:

> A special situation is presented by architectural works. An architect's plans and drawings would, of course, be protected by copyright, but the extent to which that protection would extend to the structure depicted would depend on the circumstances. Purely nonfunctional or monumental structures would be subject to full copyright protection under the bill,

589. *See* H.R. REP. NO. 94-1476, at 55 (1976); Jane Ginsburg, *"Courts Have Twisted Themselves into Knots": US Copyright Protection for Applied Art*, 40 COLUM. J.L. & THE ARTS 1 (2016); Shira Perlmutter, *Conceptual Separability and Copyright in the Designs of Useful Articles*, 37 J. COPYRIGHT SOC'Y U.S.A. 339, 351 (1990) (explaining that the House Judiciary Committee "stressed Congress's desire to exclude from protection the general class of industrial products, notwithstanding any 'aesthetically satisfying' design").

590. *See* H.R. REP. NO. 94-1476, at 55 (1976).

> and the same would be true of artistic sculpture or decorative ornamentation or embellishment added to a structure. On the other hand, where the only elements of shape in an architectural design are conceptually inseparable from the utilitarian aspects of the structure, copyright protection for the design would not be available.[591]

As part of achieving compliance with the Berne Convention, Congress amended the Copyright Act in 1990 to expand protection for architectural works and move away from the separability standard for architectural works.[592] Nonetheless, Congress retained non-functionality as a threshold requirement for copyrightability. The Architectural Works Copyright Protection Act defined "architectural work" to include the "the overall form as well as the arrangement and composition of spaces and elements in the design, but does not include individual standard features."[593] Thus, Congress retained a threshold exclusion for "individual standard features."[594] The legislative history explains that

> The Committee does not suggest . . . that in evaluating the copyrightability *or* scope of protection for architectural works, the Copyright Office or the courts should ignore functionality. A two-step analysis is envisioned. First, an architectural work should be examined to determine whether there are original design elements present, including overall shape and interior architecture. If such design elements are present, a second step is reached to examine whether the design elements are functionally required. If the design elements are not functionally required, the work is protectible without regard to physical or conceptual separability.[595]

---

591. *See* H.R. REP. NO. 1476, 94th Cong., 2d Sess. at 55.

592. *See* Architectural Works Copyright Protection Act, Pub. L. No 101-650, 104 Stat. 5089 (1990) (codified at 17 U.S.C. §§ 101, 102(a)(8), 120); H.R. REP. NO. 101-735, *as reprinted in* 1990 U.S.C.C.A.N. 6935, 6952.

593. 17 U.S.C. § 101 (defining "architectural work").

594. *See id.*; H.R. REP. NO. 101-735, *as reprinted in* 1990 U.S.C.C.A.N. 6935, 6949 (explaining that "the definition makes clear that protection does not extend to individual standard features, such as common windows, doors, and other staple building components").

595. H.R. REP. NO. 101-735, Copyright Amendments Act of 1990, 1990 U.S.C.C.A.N. 6935, 6951–52 (emphasis added).

Thus, Congress plainly intended and envisioned that courts would consider functionality in evaluating copyrightability *and* scope of protection.

Moreover, the text and legislative history of the Copyright Act, drawing on *Baker v. Selden* and its progeny, make clear that Congress intended a parsimonious approach to copyright protection for useful articles and other functional works.[596] The Federal Circuit's expansive approach — reviving the discredited *Whelan* approach and *Apple v. Franklin* dicta — contradicts the important channeling function of the idea-expression dichotomy. By bestowing copyright protection on functional specifications for a software interface whenever there are multiple ways of achieving a high-level purpose, the Federal Circuit allows copyright protection to control access to platforms or any other particular machine. This undermines the logic of the intellectual property system. Only a patented invention — which is subject to the substantial, innovation-focused threshold requirements of novelty, non-obviousness, and disclosure and limited duration — can provide such protection. By contrast, copyright protection for particular implementations affords platform developers protection against software piracy while fostering competition within non-patented platforms. Furthermore, by keeping platform specifications proprietary, developers can gain lead-time due to the difficulty of reverse engineering and hence limited control over their platform.

ii. Misreading Ninth Circuit Jurisprudence

Beyond misconstruing § 102(b), the Federal Circuit's opinion diverges from the clear language and evolution of the Ninth Circuit's software copyright jurisprudence. Judge Alsup drew principally from the First Circuit's *Lotus* decision and the Ninth Circuit's *Sega* decision in framing his analysis. The Federal Circuit held that the Lotus decision is "inconsistent" with Ninth Circuit precedent[597] and that the *Sega* decision is inapt.[598] Neither of these interpretations, however, withstands scrutiny. Furthermore, the Federal Circuit applied interpretations and analytical frameworks from Third Circuit decisions *(Apple v. Franklin* and *Whelan*) that the Ninth Circuit rejected.

a. Viability of the *Lotus* Decision in the Ninth Circuit

The Federal Circuit ruled that "the Ninth Circuit has not adopted the court's 'method of operation' reasoning in *Lotus*, and [concluded]

596. *See* H.R. REP. NO. 94-1476, at 54–55, 56–67 (1976).
597. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1365 (Fed. Cir. 2014).
598. *Id.* at 1369.

that [Lotus] is inconsistent with binding precedent."[599] While it is true that the Ninth Circuit has not expressly adopted the First Circuit's "method of operation" framework, it has never rejected it. Furthermore, the Ninth Circuit's *Sega* decision, which predates the *Lotus* decision, is consistent with its analysis.

The Federal Circuit's ruling that the *Lotus* framework is "inconsistent with binding precedent" extrapolates well beyond the holding of the Ninth Circuit's *Johnson Controls* decision. In that early decision that focused on copyright protection for computer code as opposed to API design, the Ninth Circuit held that "[w]hether the nonliteral components of a program, including the structure, sequence and organization and user interface, are protected depends on whether, on the particular facts of each case, the component in question qualifies as an expression of an idea, or an idea itself."[600] That terse opinion neither distinguishes between API design and implementing code nor addresses interoperability.

The Federal Circuit reinforces its strained reading of Ninth Circuit precedent by reference to *Atari Games v. Nintendo*,[601] its own early decision applying Ninth Circuit law, that concluded that copyright law protects "the expression of [a] process or method."[602] The Ninth Circuit has never embraced that ruling, and has conclusively held in *Sega* and *Sony* that interface specifications necessary for interoperability are not copyrightable.[603] Therefore, at least in that critical context, "the expression of a process or method" is *not* copyrightable under Ninth Circuit law.

Thus, a fairer reading of Ninth Circuit jurisprudence is that although the Ninth Circuit has not had occasion to specifically address the *Lotus* line of analysis, it holds that software that is necessary for interoperability is not copyrightable. In *Sega v. Accolade*, the Ninth Circuit states that "the functional requirements for compatibility with the Genesis [video game] console [are] aspects of Sega's programs that are not protected by copyright. 17 U.S.C. § 102(b)."[604] Such aspects of the Genesis video game platform are functional specifications of the computer system — a relatively simple API. The Ninth Circuit paralleled the *Lotus* analysis and unequivocally held that the interface specification was not copyrightable. But the Ninth Circuit could not have cited the First Circuit's *Lotus* decision because that decision was

---

599. *Id.* at 1365 (citing Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173 (9th Cir. 1989); in an accompanying footnote, the Federal Circuit notes that the Ninth Circuit had only cited the *Lotus* decision once on a procedural issue.).

600. Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989).

601. 897 F.2d 1572 (Fed. Cir. 1990).

602. Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 839 (Fed. Cir. 1992).

603. *See infra* Section IV(A)(2)(ii)(b).

604. Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992).

not handed down until several years later. The *Sega* case had an additional alleged infringement: Accolade made hundreds of intermediate copies to ascertain the unprotectable interface specification. As the next Section explains, the Federal Circuit erroneously used that entirely separate issue to disregard the Ninth Circuit's clear statement that the functional requirements for compatibility — the API specifications — are uncopyrightable.

b. Disregarding the Sega/Sony Decisions

Judge Alsup properly drew heavily upon the Ninth Circuit's *Sega* decision,[605] reaffirmed in *Sony v. Connectix*,[606] for the proposition that the code required for interoperability of computer systems is uncopyrightable.[607] The Federal Circuit downplayed the relevance of these decisions based on its characterization that both "are fair use cases in which copyrightability was addressed only tangentially."[608] The Federal Circuit further rejected "Google's suggestion that *Sony* and *Sega* created an 'interoperability exception' to copyrightability."[609]

As suggested above, a careful reading of the *Sega* decision, as *Sony v. Connectix* reaffirmed, contradicts the Federal Circuit's characterization. While it is true that both cases addressed fair use issues, it was necessary to address fair use only because Sega and Sony had not made their APIs publicly available. As a result, the defendants (Accolade and Connectix) needed to make numerous copies of the entire software programs in order to reverse engineer the pertinent APIs. None of that, however, detracts from or downplays the Ninth Circuit's clear antecedent ruling: that the code necessary for interoperability was uncopyrightable.

The fair use ruling is merely icing on the pro-interoperability/pro-functionality cake. It expands the safe harbor for using API specifications necessary for interoperability by authorizing repeated copying of the entirety of computer programs — including the copyright-protected aspects — for purposes of determining the unprotectable elements. The underlying cake (uncopyrightability of interface specifications needed for interoperability or achieving a particular function) is not the least bit "tangential" to the Ninth Circuit's rulings. It is foundational to these decisions. Had Sun not made the Java APIs publicly available, Google could have copied the full Java platform software (potentially hundreds of times) to determine the unprotectable

605. *See id.* at 1510.

606. 203 F.3d 596 (9th Cir. 2000).

607. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 1000 (N.D. Cal. 2012) (characterizing the *Sega* and *Sony* cases as "close analogies" to the *Oracle v. Google* case).

608. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1369 (Fed. Cir. 2014).

609. *Id.* at 1370.

APIs. But that in no way alters the uncopyrightability of the API elements necessary for interoperability.

The last point is critical to understanding the importance of the Ninth Circuit's *Sega* and *Sony* decisions. Both decisions expressly hold that the software code necessary for interoperability is unprotectable by copyright law. These holdings are essential to the Ninth Circuit analysis. This is entirely consistent with the CONTU Report and § 102(b) of the Copyright Act. It also shows that the Ninth Circuit recognizes an "interoperability exception" to copyrightability so long as the second-comer independently re-implements the functional specifications.

The Federal Circuit attempts to rebut this reading by suggesting that it "contradict[s] Ninth Circuit case law recognizing that both the literal and non-literal components of a software program are eligible for copyright protection. And it would ignore the fact that the Ninth Circuit endorsed the abstraction-filtration-comparison inquiry in *Sega* itself."[610] This reasoning misses the mark for several reasons. First, the *Johnson Controls* case did not focus on APIs but rather the entirety of a sophisticated computer program.[611] Second, the *Johnson Controls* decision does not delve into the specific program features. The Ninth Circuit was reviewing the grant of a preliminary injunction under the "limited" abuse of discretion standard.[612] The court had substantial evidence that the defendants copied many elements of the software program in question.[613] Third, the Ninth Circuit use of the abstraction-filtration-comparison test for analyzing the copyrightability of computer code in no way contradicts the uncopyrightability of functional or network features of computer systems. Furthermore, the *Sega* case comes after *Johnson Controls* and provides a clear, well-reasoned analysis of why code necessary for interoperability is uncopyrightable. The CONTU REPORT could not be more clearer on this point: "In the computer context [the idea-expression dichotomy] means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement."[614] To achieve the same particular functionality of the 37 Java API packages, Google had to copy the precise declarations of those APIs. They

610. *Id.* at 1370 (citing Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989)).

611. *See supra* text accompanying notes 91–95.

612. *See Johnson Controls*, 886 F.2d at 1174 ("Our review of a preliminary injunction is limited. We will reverse the granting of a preliminary injunction only if the district court abused its discretion, or based its decision on an erroneous legal standard or clearly erroneous findings of fact. Dumas v. Gommerman, 865 F.2d 1093, 1095 (9th Cir. 1989).").

613. *Id.* at 1175–76 ("The special master's report sets forth, in detailed form, the various similarities between the programs.").

614. *See* CONTU REPORT, *supra* note 47, at 20 (footnote omitted).

are the equivalent of the "functional requirements for compatibility with the Genesis [video game console] — aspects of Sega's programs that are not protected by copyright."[615]

c. Resurrecting the Third Circuit's *Apple*/*Whelan* Decisions

Not only does the Federal Circuit misread the Ninth Circuit's *Sega* and *Sony* decisions, it embraces lines of analysis that the Ninth Circuit rejected. By holding that the code for interoperability may be protectable, the Federal Circuit resurrects the Third Circuit's dicta in *Apple v. Franklin*: "courts have recognized that, once the plaintiff creates a copyrightable work, a defendant's desire 'to achieve total compatibility . . . is a commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged.'"[616] To the contrary, the Ninth Circuit holds that copyright law does not stand in the way of achieving functional interoperability. As noted earlier,[617] the Third Circuit comment is dicta as Franklin Computer had copied the entirety of Apple's computer programs. More importantly, § 102(b), the CONTU REPORT, and the *Sega*/*Sony* decisions directly contradict the Third Circuit's proposition.

The Federal Circuit endorses and follows the Third Circuit's *Apple/Whelan* framework, holding that everything not necessary to the *general* purpose or function of a work is protectable expression: "We agree with Oracle that, under Ninth Circuit law, an original work — even one that serves a function — is entitled to copyright protection as long as the author had multiple ways to express the underlying idea."[618] The Federal Circuit credited Oracle's statement that it only claimed "its *particular* way of naming and organizing each of the 37 Java API packages" and that it "'cannot copyright the idea of programs that open an internet connection,' but 'it can copyright the precise strings of code used to do so, at least so long as "[another] language is available" to achieve the same function.'"[619] In an accompanying footnote, the court noted that Oracle's counsel explained at oral argument that Oracle "would never claim that anyone who uses a

615. Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992) (citing 17 U.S.C. § 102(b)).

616. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1357 (Fed. Cir. 2014).

617. *See supra* notes 83–86.

618. *See Oracle v. Google*, 750 F.3d at 1371 (quoting Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983)); *see also id.* at 1366 (noting that the Third Circuit in *Apple v. Franklin* "focused 'on whether the idea is capable of various modes of expression' and indicated that, '[i]f other programs can be written or created which perform the same function as [i]n Apple's operating system program, then that program is an expression of the idea and hence copyrightable'" (quoting *Apple v. Franklin*, 714 F.2d at 1252)).

619. *See Oracle v. Google*, 750 F.3d at 1367–68 (emphasis in original; internal quotations from Oracle's Reply Brief).

package-class-method manner of classifying violates our copyright. We don't own every conceivable way of organizing, we own only our specific expression — our specific way of naming each of these 362 methods, putting them into 36 classes, and 20 subclasses."[620] The Federal Circuit reasoned that if different code could perform the same general functions, then the first author's code for such general functions was protectable.[621]

While this mode of analysis comports with Ninth Circuit jurisprudence on code implementation, it contradicts copyright law principles and Ninth Circuit precedent regarding the declarations that are necessary to operate a particular computing system. Contrary to the Third Circuit's dicta in *Apple v. Franklin*, the Ninth Circuit's *Sega* and *Sony* decisions hold that the code necessary for interoperability is uncopyrightable.[622] Thus, a defendant's desire to achieve compatibility *does* enter into the issue of whether particular ideas and expressions have merged in the Ninth Circuit. It resolves the issue so long as the defendant independently writes the code to achieve the particular functions of the plaintiff's software. Secondly, the *Sega* decision unequivocally rejects the *Whelan* framework of simply asking whether there are multiple ways of programming a particular function: "[t]he *Whelan* rule . . . has been widely — and soundly — criticized as simplistic and overbroad."[623]

The Federal Circuit elides this issue by emphasizing that the Ninth Circuit adopted the *Altai* abstraction-filtration-comparison framework. But the Ninth Circuit's endorsement of the *Altai* framework for cases involving implementing code does not exclude the

---

620. *Id.* at 1367 n.13.

621. *See id.* at 1356 (setting the foundation for its analysis by observing that "the Sun/Oracle developers had a vast range of options for the structure and organization" of the Java APIs); *id.* at 1360 ("We have recognized , . . . applying Ninth Circuit law, that the 'unique arrangement of computer program expression . . . does not merge with the process so long as alternate expressions are available.'" (quoting Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 840 (Fed. Cir. 1992))); *id.* (explaining that "[b]ecause Nintendo produced expert testimony 'showing a multitude of different ways to generate a data stream which unlocks the NES console,' we concluded that Nintendo's specific choice of code did not merge with the process."); *id.* at 1360 n.5 (noting that "[i]t is undisputed that Microsoft and Apple developed mobile operating systems from scratch, using their own array of software packages."); *id.* at 1368 n.14 (referencing the amicus brief of former Sun executives explaining that "a quick examination of other programming environments [Apple's iOS and Microsoft Windows Phone] shows that creators of other development platforms provide the same functions with wholly different creative choices.").

622. *See* Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1522 (citing 17 U.S.C. § 102(b)) (holding that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright.").

623. *See Sega*, 977 F.2d at 1522 (citing Comput. Assocs. Int'l, Inc. v. Altai, 1992 WL 139364 (2d Cir. June 22, 1992), *opinion withdrawn and superseded on rehearing by* 982 F.2d 693 (2d Cir. 1992)). It recognized that "the functional requirements for compatibility with [a software platform developed by another company] are not protected by copyright." *Id.* at 1522 (citing 17 U.S.C. § 102(b))).

possibility that the Ninth Circuit would not apply § 102(b) and the *Lotus* decision in analyzing copyrightability of API design elements that are essential to achieving a particular set of functions. In fact, as Judge Alsup recognized, the *Sega* ruling supports such as an approach. In any case, the Federal Circuit contradicted Ninth Circuit precedent in following the Third Circuit's *Apple v. Franklin* and *Whelan* decisions.

iii. Conflation of Expressive and Technological "Creativity"

The Federal Circuit embraced Oracle's argument (and that of former Sun executives)[624] that API design is a "creative," "noble and rewarding craft"[625] entitled to robust protection. Oracle analogized API design to the crafting of HARRY POTTER novels.[626]

Plaintiffs in the first wave of API copyright litigation deployed a similar strategy,[627] but, as the courts came to appreciate, analogizing software design to classical operas overlooks the fundamental principles undergirding the intellectual property system. For purposes of copyright law, technological creativity differs fundamentally from expressive creativity. Section 102(b) provides the touchstone. Where the work or element of the work constitutes an "idea, procedure, process, system, method of operation, concept, principle, or discovery," no amount of artistic or expressive "creativity" will suffice. The work or element of a work falls outside of copyright protection. That is the point of *Baker v. Selden* and the idea-expression dichotomy. "[T]he actual processes or methods embodied in [a computer] program are not within the scope of the copyright law."[628] "[O]ne is always free to make a machine perform any conceivable process (in the absence of a patent)" so long as they don't "take another's program . . . . [W]hen specific instructions . . . are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement."[629]

The Federal Circuit, however, accepted Oracle's analogy between complex API design and the crafting of HARRY POTTER novels.[630]

---

624. *See* Corrected Brief of Scott McNealy & Brian Sutphin as Amici Curiae in Support of Reversal, Oracle Am., Inc. v. Google Inc., No. 2013-1021, -1022 (Fed. Cir. filed Feb. 22, 2013) (characterizing API design as a highly creative process in which programmers select names from a limitless pallet of choices).

625. *See* Opening Brief and Addendum of Plaintiff-Appellant, Oracle Am., Inc. v. Google Inc., No. 2013-1021, -1022, 12-13, 72 (Fed. Cir. Feb. 11, 2013).

626. *See supra* text accompanying note 436.

627. *See supra* note 90.

628. H.R. REP. NO. 94-1476, at 66 (1976).

629. *See* CONTU REPORT at 20.

630. *See Oracle v. Google*, 750 F.3d at 1356 (citing the district court's copyrightability decision for the proposition that "[t]he overall name tree [of the Java API, of course, has creative elements").

The court portrays Java's APIs as highly creative, difficult, and time-consuming works of authorship:

> Scott McNealy and Brian Sutphin — both former executives at Sun who were involved in the development of the Java platform — provide a detailed example of the creative choices involved in designing a Java package. Looking at the 'java.text' package, they explain that it 'contains 25 classes, 2 interfaces, and hundreds of methods to handle text, dates, numbers, and messages in a manner independent of natural human languages.' Java's creators had to determine whether to include a java.text package in the first place, how long the package would be, what elements to include, how to organize that package, and how it would relate to other packages. This description of Sun's creative process is consistent with the evidence presented at trial. *See* Appellant Br. 12–13 (citing testimony that it took years to write some of the Java packages and that Sun/Oracle developers had to 'wrestle with what functions to include in the package, which to put in other packages, and which to omit entirely').[631]

There is no question that both J.K. Rowling's novels and sophisticated API designs are "creative" in a dictionary sense of the term.[632] The critical distinction, however, relates to the idea-expression dichotomy and the channeling of protection among the modes of intellectual property law. Affording robust protection to the characters, settings, and plot elements of J.K. Rowling's stories does not monopolize book technology or literary communication. Furthermore, subsequent authors remain free to develop their own wizardry stories. They may freely partake of Rowling's ideas, just not her expression of those ideas.

By contrast, APIs function as the levers and gears of particular digital machines. The declarations must be reproduced to replicate the *particular* functionality. As shown in Figure 10, Android programmers needed to reproduce the same package name (java.security), class name (ProtectionDomain), and method name (ClassLoader) to effectuate a computer program that responds to the same inputs and produces the same outputs as the java.security machine. As the faded

631. *Oracle v. Google*, 750 F.3d at 1361 n.6 (some citations omitted).

632. *See* WEBSTER'S THIRD NEW INTERNATIONAL DICTIONARY 532 (1986) (defining "creative" as "having the quality of something created rather than imitated or assembled").

background text indicates (and as Oracle acknowledged), Google implemented the declarations using different code.

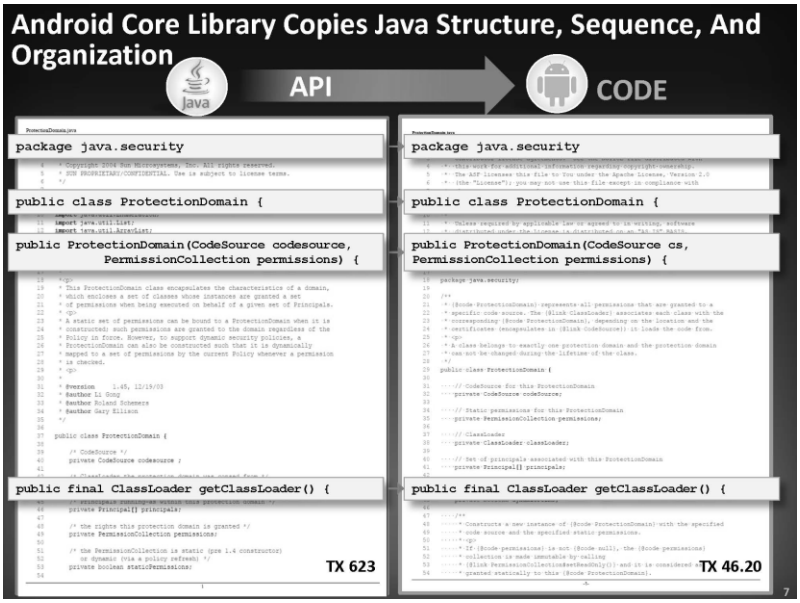

Figure 10. Oracle's Closing Argument Slide Deck, Slide 7
(Figure 4 repeated)

To a lay audience, phrases such as "ProtectionDomain" and "getClassLoader" as well as mingling of traditional parentheses and curly braces could indicate arbitrary, if not creative, expression. To the Java API developer and the third-party programmers seeking to invoke particular Java APIs, the names, along with capitalization, define the particular methods that the virtual machine runs. The curly braces signify that source code follows the declarations (definitions). Their usage follows the syntactical rules of the Java programming language. Hence, it is no coincidence that Android contains the identical declarations as the Java APIs.

As reflected in Figure 11, which sets forth the official Java specification of the ProtectionDomain class,[633] a component of the java.security API, Android programmers copied the method names necessary to effectuate the java.security functionality. The particular

633. *See java.security, Class ProtectionDomain*, JAVA™ PLATFORM, STANDARD EDITION 7 API SPECIFICATION, ORACLE, https://docs.oracle.com/javase/7/docs/api/java/security/ProtectionDomain.html [https://perma.cc/8CVR-SK9S]; *API Specification*, JAVA™ PLATFORM, STANDARD EDITION 7 API SPECIFICATION, ORACLE, https://docs.oracle.com/javase/7/docs/api/overview-summary.html [https://perma.cc/9CER-VV7T].

combination of methods and constructors defines the ProtectionDomain sub-machine. Diverging from a single character (or capitalization) within a declaration renders the particular machine inoperable. This is no different from having the wrong activation code for the Sega Genesis platform or using the wrong letters in a Lotus macro, or the wrong PIN for an ATM.

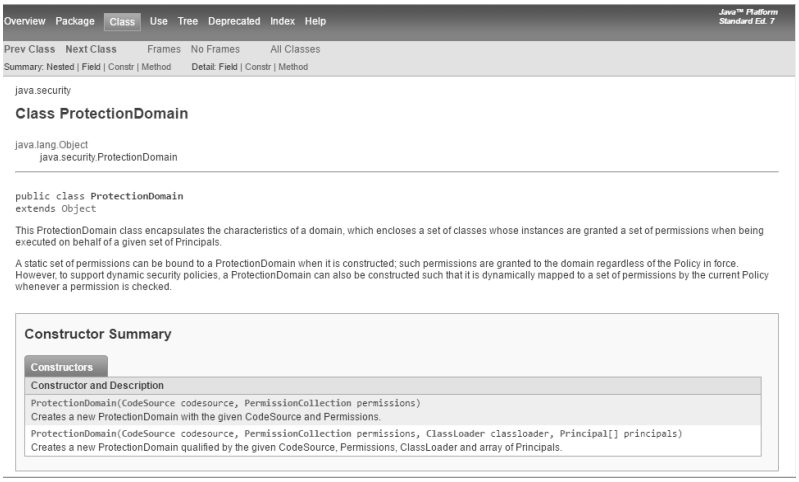

Figure 11. Java Platform Standard Edition 7: Package java.security, Class ProtectionDomain

Affording copyright protection for a combination of methods and constructors monopolizes that particular machine or sub-machine. It is for this reason that Congress, through § 102(b), and the Supreme Court, through *Baker v. Selden*,[634] require inventors to meet utility patent law's higher thresholds of novelty, non-obviousness, and disclosure to garner protection for technological innovation. Furthermore, that protection is limited to 20 years from the filing of the application. Such limitations promote competition and innovation. Consumers do not become locked into arbitrary and only minimally innovative platforms and competitors have greater freedom to expand and improve platforms.

The Federal Circuit's decision misses the profound wisdom of *Baker v. Selden* and § 102(b). The panel's conflation of expressive and technological creativity allows Oracle and other platform sponsors to control access to technological platforms for what is effectively eternity (95 years for corporate authors) merely by satisfying copyright law's low originality threshold for protection so long as

634. 101 U.S. 99 (1879).

there are multiple ways of achieving the same *general* functionality of the platform (which there usually is). To the ten million Java programmers, having to learn a new programming platform is a substantial impediment to making the switch to what may be a better platform. It destroys their substantial investment in human capital and code development. It was that same consideration that made it commercially important for Borland to implement a feature that would enable macros designed to run on Lotus 1-2-3 to operate on Borland's Quattro.

The Federal Circuit's analysis is internally inconsistent. In distinguishing the *Lotus* case, the Federal Circuit explains that "the *Lotus* court found that the commands at issue there (copy, print, etc.) were not creative, but it is undisputed [in *Oracle v. Google*] that the declaring code and the structure and organization of the API packages are both creative and original."[635] Yet earlier in its opinion, the Federal Circuit explained that the Lotus "menu command hierarchy referred to a series of commands — such as 'Copy,' 'Print,' and 'Quit' — which were arranged into more than 50 menus and submenus."[636] If, as the Federal Circuit recognized, the creativity threshold for copyright protection is "not particularly stringent,"[637] then the unique selection and arrangement of hundreds of labels easily clears copyright's creativity threshold.[638] Furthermore, there are many alternative labels that could be selected, such as "Reproduce" for "Copy" and "End" for "Quit." The difference between the First Circuit's and the Federal Circuit's analyses is that the First Circuit recognized that the technological creativity involved in API design is not of the type that copyright protects and only utility patent law can protect machine functionality.

Proper application of the idea-expression dichotomy solves the creativity conundrum. Menu command hierarchies and API designs are technological and hence uncopyrightable — they are ideas, processes, systems, and methods of operation. They might also be mathematical discoveries, which fall within the constitutional schema for utility patent protection. The implementing code for these designs, however, is copyrightable. Java API declarations are functional specifications that are necessary to achieve *particular* functionality. Hence, they fall outside of copyright protection. Only those particular segments of implementing code that are necessary to accomplish a par-

635. *Oracle v. Google*, 750 F.3d at 1365.

636. *Id.*

637. *See id.* at 1354 (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 358 (1991)); *see also Feist*, 499 U.S. at 354 (characterizing copyright's creativity threshold as "low").

638. *See* 17 U.S.C. § 103(a) (extending copyright protection to "compilations"); *id.* at 101 (defining a "compilation" as "a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship").

ticular function (i.e., merger) as well as those structural features that are unoriginal (e.g., copied from others, standard programming techniques (*scènes à faire*)) or functionally-based (e.g., the most efficient means of coding) are free for others to use.

Thus, for purposes of copyright law, technological creativity differs fundamentally from expressive creativity. Where a work or element of a work constitutes an "idea, procedure, process, system, method of operation, concept, principle, or discovery,"[639] no amount of creativity will suffice to attract copyright protection. That work or element of that work still falls outside of copyright protection.

iv. Overly Rigid Approach to Limiting Doctrines

The Federal Circuit errs by shoehorning analysis of API design into a framework designed for analyzing copying of software code. As the *Lotus* court and Judge Alsup recognized, copyright law does not dictate a monolithic approach. Copyright law has long relied upon a common-law approach for adapting the law to deal with new technologies and other dynamic considerations.[640] The idea-expression dichotomy provides flexibility in the domain of functional works. Courts need to be sensitive to the technological nuance in applying § 102(b) and evolving the family of doctrines (idea-expression dichotomy, merger, *scènes à faire*, *Baker v. Selden*, and fair use) on which it is based.

I confronted a similar challenge two decades ago. As I noted earlier,[641] I co-organized an effort in the late 1980s to identify consensus regarding how copyright law should treat computer software. My colleagues and I were concerned that several of the early software copyright decisions misunderstood the nature of computer programming and the economics of software markets. We convened leading copyright authorities to discuss how traditional copyright principles, most notably the idea-expression dichotomy and fair use, should apply to computer software. Our consensus report articulated several principles and ways of addressing particular issues. Among our recommendations was that the fair use doctrine provided a pathway for software developers to reverse engineer computer programs to determine how they worked. We also believed that programmers should be free to develop interoperable computer programs so long as they wrote their own implementations.

639. 17 U.S.C. § 102(b).
640. *See* Menell, *supra* note 569, at 70.
641. *See supra* note 5.

Our work was cited in the parties' and amici briefs filed in the *Altai* case.[642] That litigation produced a watershed in the evolution of software copyright doctrine.[643] When the *Sega* case emerged, we saw an opportunity to bring our insights to the attention of the Ninth Circuit. Our amicus brief articulated the framework that the court adopted.[644]

As the *Lotus* case headed for appeal, a group of copyright law professors circulated a brief arguing that the *Altai* framework provided the key to correcting Judge Keeton's district court decision.[645] While I applauded the abstraction-filtration-comparison framework as a sound basis for analyzing the alleged infringement of computer code, I did not feel that this approach fit the *Lotus* case. Borland had not copied the Lotus code but rather had independently implemented the menu command hierarchy — an API design. The particular structure and function names were required to run macros written for the Lotus platform. Professor Dennis Karjala and I saw copyrightability, as opposed to infringement analysis, to be the proper framework for addressing the *Lotus* case.[646] We emphasized the underlying principle

---

642. *See, e.g.*, Brief Amicus Curiae of American Committee for Interoperable Systems, Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 696 (2d Cir. 1992) (No. 91-7893) (1991 WL 11010231).

643. While the district court and Second Circuit were receptive to our work, advocates for broad copyright protection for computer software were critical. *See* Anthony L. Clapes, *Confessions of an Amicus Curiae: Technophobia, Law and Creativity in the Digital Arts*, 19 U. DAYTON L. REV. 903, 923 (1994). Mr. Clapes, then-Assistant General Counsel at IBM, which had acquired Lotus Corporation, noted that "[t]he [*Altai*] court cited only one law review article and one academic text as sources of criticism of the Third Circuit rule that a program's structure, sequence, and organization may be protectable expression. The law review article was written by a well-known antiprotectionist law professor." The accompanying footnote states: "In addition to being a member of the widely criticized LaST Frontier conference steering committee, Professor Menell is a member of the 'gang of ten' law professors who filed amicus briefs in support of copyright defendants in software copyright cases." *See id.* at 923 n.81; *see also id.* at 913 n.23 ("Perhaps unaware of the peculiar Luddist filter through which Professor Menell looks at the art of programming, the [*Altai*] court adopted his views as to the nature of computer programs in whole cloth."). As the LaST Frontier Conference explains, we sought to achieve balance among the law professors at the conference and included a wide range of perspectives. *See LaST Frontier Software Report*, *supra* note 5, at 15. Mr. Clapes co-authored the highly imaginative and misleading article analogizing computer programs to literature and opera. *See* Anthony L. Clapes, Patrick Lynch & Mark R. Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493 (1987) (authored by in-house and outside counsel for IBM). I did not represent parties in any of this work and was not compensated for submitting any of these briefs.

644. *See* Brief Amicus Curiae of Copyright Law Professors, Sega Enters. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

645. *See generally* Brief Amicus Curiae of Copyright Law Professors at 33, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995) (No. 93-2214) (1993 WL 13624511) (arguing that "The Successive Filtering Test for Infringement Endorsed in *Altai* Is More Consistent With Traditional Principles of Copyright Law Than Is The *Paperback/Borland* Test").

646. *See* Brief Amicus Curiae of Professor Dennis S. Karjala & Professor Peter S. Menell, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995) (No. 93-2214) (1993

of *Baker v. Selden* as well as the language of § 102(b) categorically excluding menu command hierarchies from copyright protection. Similarly, the First Circuit properly distinguished the *Altai* case as dealing with software code as opposed to API design.[647]

Oracle v. Google is the first litigated copyright case since *Lotus* to focus specifically on copyright protection for API design.[648] Judge Alsup saw that although the Ninth Circuit had endorsed the *Altai* framework for cases involving implementing code, the *Oracle v. Google* case required an alternative framework to address API design. He recognized that the *Lotus* case provided pertinent analysis and that the *Sega* case addressed the uncopyrightability of code necessary for interoperability. His decision thoughtfully combined these elements to produce a sound framework.[649]

Beyond the misreading of copyright law and Ninth Circuit jurisprudence, the Federal Circuit's decision elevates wooden rules over

---

WL 13624512), *reprinted in* Dennis S. Karjala & Peter S. Menell, *Applying Fundamental Copyright Principles to* Lotus Development Corp. v. Borland International, Inc., 10 HIGH TECH. L.J. 177 (1995).

647. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995) (recognizing that:

> While the *Altai* test may provide a useful framework for assessing the alleged nonliteral copying of computer code, we find it to be of little help in assessing whether the literal copying of a menu command hierarchy constitutes copyright infringement. In fact, we think that the *Altai* test in this context may actually be misleading because, in instructing courts to abstract the various levels, it seems to encourage them to find a base level that includes copyrightable subject matter that, if literally copied, would make the copier liable for copyright infringement. While that base (or literal) level would not be at issue in a nonliteral-copying case like Altai, it is precisely what is at issue in this appeal. We think that abstracting menu command hierarchies down to their individual word and menu levels and then filtering idea from expression at that stage, as both the *Altai* and the district court tests require, obscures the more fundamental question of whether a menu command hierarchy can be copyrighted at all. The initial inquiry should not be whether individual components of a menu command hierarchy are expressive, but rather whether the menu command hierarchy as a whole can be copyrighted.

(footnote and citation omitted)).

648. As noted above, the *Sega* case addressed this issue as part of a fair use analysis of intermediate copying of software code. *See supra* notes 122–34. This API design issue has, however, arisen in other litigation contexts, but was not resolved by judicial decisions. As noted earlier, *see supra* note 17, I advised Sun Microsystems about these issues during their Java-related breach of contract and copyright infringement litigation with Microsoft in the late 1990s. I also testified about these issues in an arbitration proceeding applying Ninth Circuit law in 2007. *See supra* note 20. The arbitration panel interpreted Ninth Circuit law very similarly to Judge Alsup and found the declaring code (header files) at issue in that case to be uncopyrightable. (I served as a consultant in the Sun matter and as an expert witness in the Green Hills matter. In contrast to my work on amicus briefs, I was compensated by the parties that retained me in these matters. My testimony reflected my writings on legal protection for computer software.)

649. *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 984–97 (N.D. Cal. 2012), rev'd and remanded, 750 F.3d 1339 (Fed. Cir. 2014).

the underlying logic of copyright law. The Federal Circuit builds much of its copyrightability analysis on the notion that merger analysis must be evaluated at the time that the plaintiff's work is created, not at the time that the defendant prepares its allegedly infringing work. It also brings copyright's limiting doctrines into play as part of a fair use defense.[650] The Federal Circuit supports this proposition by noting that the CONTU REPORT recognized that the Copyright Act was designed "to protect all works of authorship from the moment of their fixation in any tangible medium of expression."[651] While accurate in describing § 102(a), the statement ignores the comparably important limiting doctrines of § 102(b). As the CONTU REPORT recognized later in that same paragraph, the Copyright Act leaves application of the idea-expression doctrine to the judgment of the courts.[652]

The Federal Circuit's decision misses a fundamental difference between conventional literary works that serve to engage, amuse, and entertain readers and computer code that serves to operate particular computer devices. When Sega developed its lock-out code for the Genesis game console, there were no constraints on the arbitrary string characters that it designated for the key. Just as bank customers are unconstrained in choosing their PIN codes (within the field constraint of four numbers), Sega was free to choose an arbitrary string of letters, numbers, and symbols to lock and unlock its platform. Yet the Ninth Circuit determined that the lock-out code was unprotectable under § 102(b) because once it was "created" for use as lock-out code, it became functional.

The First Circuit reached a similar conclusion in the *Lotus* case. At the time that Lotus designed its menu command hierarchy for the Lotus 1-2-3 program, there were numerous options for labeling the functions and countless compilations of function names. Once programmers learned these function names, however, they become important to the user community. To bestow copyright protection on such a system would potentially confer tremendous market power over the particular method of operating a spreadsheet due to users' high switching costs — many had developed sophisticated macros for automating their accounting and other record keeping. The First Circuit recognized that this issue was best addressed at the copyrightability stage. Like Selden's accounting book, Lotus's spreadsheet program was entitled to copyright protection at the moment it was created (or in the case of Selden's book, when the applicable formali-

650. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1361 (Fed Cir. 2014).
651. *See id.* (quoting CONTU REPORT at 21).
652. *See* CONTU REPORT at 21.

ties at the time were met) but the method of operation (like Selden's accounting system) remained outside of copyright protection.

Although far more sophisticated than an ATM PIN code, the Genesis lock-out code, or even Lotus's multi-level menu command hierarchy, the declarations of the Java APIs similarly functioned as methods of operating particular digital machine — packages of functions. Judge Alsup's focus on § 102(b) and the *Lotus* court's framework better addresses the copyright issues in *Oracle v. Google* than the *Altai* framework, which was developed for analyzing copyright code.

By rigidly focusing on Ninth Circuit cases that treat the merger and *scènes à faire* doctrines as defenses to infringement rather than copyrightability doctrines,[653] the Federal Circuit missed the forest for the trees. Section 102(b) can operate as both a threshold doctrine and as part of the filtration step of infringement analysis. In fact, in the *Ets-Hokin* case, on which the Federal Circuit bases its analysis, the Ninth Circuit treats the bottle that is the object of the photograph in question as uncopyrightable under the useful article doctrine, at a threshold copyrightability level.[654] Copyright law, like patent law's non-obviousness doctrine, does not fit a rigid mold.[655]

v. Treating API Design as Variable Expression Rather than Unique Function

The root cause of the Federal Circuit's flawed analysis is its treatment of the set of thirty-seven Java API declarations — what it confusingly called declaring "code" — as software code rather than as the essential functional specifications for a computer system.[656] Such

---

653. The Federal Circuit cites to *Ets-Hokin v. Skyy Spirits, Inc.*, 225 F.3d 1068, 1073, 1082 (9th Cir. 2000) (involving photography) and *Satava v. Lowry*, 323 F.3d 805, 810 n.3 (9th Cir. 2003) (involving glass-encased jellyfish sculptures; holding that "[t]he Ninth Circuit treats scènes à faire as a defense to infringement rather than as a barrier to copyrightability").

654. *See* Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1073, 1080 (9th Cir.2000).

655. *Cf.* KSR Int'l. Co. v. Teleflex Inc., 550 U.S. 398 (2007) (reversing the Federal Circuit for applying too rigid a test (the teaching-suggestion-motivation requirement) for analyzing patent law's non-obviousness doctrine); *see generally* Rochelle Cooper Dreyfuss, *In Search of Institutional Identity: The Federal Circuit Comes of Age*, 23 BERKELEY TECH. L.J. 787, 802–04 (2008) (observing that "[i]nstead of scrutinizing substantive outcomes, [the Federal Circuit] began to insist on particular analytical approaches — for example, for non-obviousness, on the use of the "teaching, suggestion or motivation" test in addition to mandatory attention to secondary considerations (such as commercial success); for claim construction, it has experimented with rigid interpretative methodologies and specific forms of evidence"); John R. Thomas, *Formalism at the Federal Circuit*, 52 AM. U. L. REV. 771, 773 (2003) (noting the Federal Circuit's growing inclination toward bright line as opposed to balancing tests)).

656. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1368 (Fed Cir. 2014) ("Given the [district] court's findings that the SSO is original and creative, and that the declaring code could have been written and organized in any number of ways and still have achieved

API design defines the particular data processing capabilities of a particular computing machine and is necessary for another virtual machine to perform the same processes.

From a copyright standpoint, the critical question is whether a particular set of instructions, expressed in a particular way, is "the only and essential means of accomplishing a given task."[657] Alternatively, are these particular instructions, expressed in this particular way, the only way to effectuate "the actual processes or methods embodied in the program"?[658] As CONTU explained, "one is always free to make a machine perform any conceivable process (in the absence of a patent)" so long as they don't "take another's program."[659] The test is not whether there are multiple ways of writing code to perform a *general* purpose. Congress viewed the idea-expression dichotomy as enabling anyone to build a machine capable of performing any *particular* function, including those for which others had written computer code. Under the idea-expression dichotomy, copyright protection must not lock competitors out of a particular platform; only utility patent protection can. Copyright protection can only require that competitors write their own implementing code. If the only way to achieve such "certain result"[660] includes literally copying even detailed textual-represented information — such as declarations — then copyright law does not stand in the way.

Google followed this path. It sought to achieve the particular functionalities of thirty-seven Java API packages. After negotiations to license the Java APIs reached an impasse, Google independently wrote its own implementing code. 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. Thus, the Federal Circuit should have affirmed Judge Alsup's copyrightability ruling and the case should have ended at that stage.

3. Proper Legal Frameworks for Analyzing Copyright Protection for Computer Software

"Computer software, by its very nature as written work intended to serve utilitarian purposes, defies easy categorization within our

---

the same functions, we conclude that Section 102(b) does not bar the packages from copyright protection just because they also perform functions.").

657. *See* CONTU REPORT at 20.

658. *See* H.R. REP. No. 94-1476, at 57 (1976).

659. *See* CONTU REPORT at 20.

660. As added in the 1980 amendments, the Copyright Act defines a "computer program" as "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101.

intellectual property system."[661] Like other useful articles, computer software combines expressive and functional elements in complex ways.

The nature of computer programming and the economics of software markets were relatively primitive when Congress brought computer software into copyright law in the mid-1970s. Congress and the CONTU Commissioners articulated general principles for ensuring that copyright protection would not trench on utility patent law's domain in protecting procedures, processes, systems, methods of operation, and discoveries. Congress tasked the courts with applying and evolving the idea-expression dichotomy to protect against unwarranted market power over advances that are subject to utility patent law's more stringent requirements and shorter duration.

Computer software comprises distinct elements or components. The idea-expression dichotomy provides the principle for distinguishing between unprotectable processes and protectable expression. After some early missteps, the courts developed a variety of effective approaches for dealing with a range of software features. The *Altai* abstraction-filtration-comparison test provides an effective framework for analyzing copying of software code. The *Sega*, *Apple v. Microsoft*, and *Lotus* cases provide valuable frameworks for dealing with software components: code segments necessary for interoperability, graphical user interface features, and menu command hierarchies.

A key issue is whether to apply the idea-expression dichotomy at the threshold copyrightability stage, as part of the infringement analysis, or as part of the fair use defense. The following Sections present an overarching framework for applying copyright law to API design elements, computer code, and other software elements.

i. API Design

The declarations of an API package serve as the functional specifications of the computer program or computer program module. They specify the function prototypes[662] of the computer program, including the function names and signature arity[663] (number of independent variables), parameter types, and return types. This information defines function names, the format and number of inputs and outputs, and syntax. If this precise textual information is essential to a machine performing specific processes or methods, then they fall outside of the

661. *See* Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, *supra* note 4, at 1046.

662. *See Function Prototype*, WIKIPEDIA (Sept. 12, 2017), https://en.wikipedia.org/wiki/Function prototype [https://perma.cc/263M-ULPQ].

663. *See Arity*, WIKIPEDIA (Aug. 3, 2017), https://en.wikipedia.org/wiki/Arity [https://perma.cc/Y3HD-9HUG].

scope of copyright protection.[664] This is not a question of merger. It is a question of whether the subject matter is categorically excluded as a process or method of operation.

The separability doctrine used to analyze copyright protection for pictorial, graphic, and sculptural ("PGS") works provides a useful model for applying the idea-expression dichotomy to API declarations. Courts must determine whether the expressive features of a PGS work — such as a ribbon-shaped bicycle rack[665] — "can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article."[666] As the House Judiciary Committee explained:

> [A]lthough the shape of an industrial product may be aesthetically satisfying and valuable, the Committee's intention is not to offer it copyright protection under the bill . . . . The test of separability and independence from 'the utilitarian aspects of the article' does not depend upon the nature of the design — that is, even if the appearance of an article is determined by esthetic (as opposed to functional) considerations, only elements, if any, which can be identified separately from the useful article as such are copyrightable.[667]

Courts determine separability at the copyrightability stage of analysis.[668] Only if there are expressive elements that are separable from the utilitarian attributes of the useful article does the court proceed to the infringement stage of analysis.

API declarations entail the same basic inquiry — can the textual information in an API header be distinguished from the utilitarian aspects of the API package? The idea-expression dichotomy means that "the actual processes or methods embodied in the program are not within the scope of the copyright law."[669] If the declarations of an independently written program must contain the precise textual formation in the plaintiff's program, then those features of the program

---

664. *See* H.R. REP. NO. 94-1476, at 57 (1976) ("Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.").

665. *See* Brandir Int'l, Inc. v. Cascade Pac. Lumber Co., 834 F.2d 1142 (2d Cir. 1987); *see also* Star Athletica, L.L.C. v. Varsity Brands, Inc., 137 S. Ct. 1002, 1008 (2017) (treating the useful article doctrine as a threshold inquiry).

666. *See* 17 U.S.C. § 101 (definition of "pictorial, graphic, and sculptural" works).

667. *See* H.R. REP. NO. 94-1476, at 55 (1976).

668. *See Brandir*, 834 F.2d at 1143–48.

669. *See* H.R. REP. NO. 94-1476, at 57 (1976).

are excluded from copyright protection. It does not matter that the declarations may be compiled or structured in a "creative" manner or in a variety of ways. Other programmers are free to copy them if they are needed to effectuate particular processes or methods, just as sculptors are free to create ribbon-shaped bicycle racks if the utilitarian features are inseparable from the expressive design elements. There is no need to engage in infringement analysis in such cases.

Oracle's comparison of API declarations to chapters in HARRY POTTER novels[670] fundamentally misapprehends the idea-expression dichotomy. The book chapter titles and topic sentences do not operate a machine. By contrast, the Java API declarations define the gears and levers of a virtual machine. Without them, no one else can create a machine that performs the same processes or methods. Oracle's suggestion that "tapp[ing] into the Harry Potter fan base" by copying book chapter titles and topic sentences[671] is analogous to copying API design misses this key point. The chapter titles and topic sentences of a novel do not control any particular machine or impinge upon technological freedom. Section 102(b) excludes processes and methods of operating machines from copyright protection, not detailed plot outlines and non-functional expressive text.

Nonetheless, § 102(b) does not categorically exclude the code implementing API declarations, although it limits some elements and affects the scope of protection for such code. The next Section explores those issues.

ii. Computer Code

The code implementing the Java APIs is copyrightable. To prevail in an infringement action, the copyright owner must prove both factual and legal copying. The requirement that the plaintiff prove that the defendant actually copied the protected work allows a defendant to avoid infringement by independently writing a computer program that will accomplish the same functions as the plaintiff's copyrighted implementation code. If, however, the defendant copies some of the plaintiff's implementing code, a court must evaluate whether the defendant's program is substantially similar to the plaintiff's protected expression.

670. *See supra* note 436.

671. *See* Opening Brief and Addendum of Plaintiff-Appellant at 12–13, Oracle Am., Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. Feb. 11, 2013) (No. 13-01021).

a. Independent Creation

Unlike patent law, which does not require any proof that a defendant knew of or copied the patentee's invention,[672] copyright law requires either direct or inferential evidence from which the court can find that actual copying occurred.[673] Thus, a defendant can successfully defend a copyright lawsuit by proving that the author of the defendant's work had no knowledge of the plaintiff's work — essentially, that the defendant's work was independently created.

The notion that a sophisticated expressive work could be independently created seems fanciful in many fields of creative endeavor. As Learned Hand whimsically posited to illustrate the doctrine, "if by some magic a man who had never known it were to compose anew Keats's Ode on a Grecian Urn, he would be an 'author,' and, if he copyrighted it, others might not copy that poem, though they might of course copy Keats's."[674]

In the realm of computer programming, however, no magic is required — just time-consuming and painstaking work. Skilled programmers with no knowledge of the implementing code of a preexisting (and copyright-protected) computer program can independently develop a computer program with the same functionality if they have the proper functional specifications. As § 102(b) and the *Sega* case hold, copyright law does not protect these functional specifications, nor any code *required* to accomplish a particular task. Consequently, independent developers can use the same functions and essential code. In this way, competitors can implement these functional elements independently without infringing copyright protection.

As noted earlier,[675] it was through such a process that IBM lost its hold on the early microcomputer industry. Phoenix Technologies successfully implemented an IBM-compatible Basic Input Output System (BIOS) using a "clean room" process.

> Phoenix's clean room approach consisted of an engineering team in Texas that examined the BIOS software documented in IBM's Technical Reference manual, and wrote a set of specifications that described how the program functioned, without including any actual examples of IBM code. These

672. *See* Jurgens v. CBK, Ltd., 80 F.3d 1566, 1570 n.2 (Fed. Cir. 1996) (characterizing patent infringement as a strict liability tort); Robert P. Merges, A *Few Kind Words for Absolute Infringement Liability in Patent Law*, 31 BERKELEY TECH. L.J. 1, 3 (2016) (explaining that "[i]t is irrelevant under current [patent] law whether the defendant actually copied the patentee's technology").

673. *See* Arnstein v. Porter, 154 F.2d 464 (2d Cir. 1946).

674. *See* Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54 (1936).

675. *See supra* note 34.

> specifications were given to a single programmer in Massachusetts who had no experience with the IBM's microprocessor. Another Phoenix employee acted as a gatekeeper to route formal questions so as to ensure that the engineers in Texas did not give the programmer in Massachusetts any material that might infringe IBM copyrights. 'A third group tested the Phoenix software against a variety of programs that ran on IBM's computer.' Phoenix engineers created an evidentiary audit trail nearly five thousand pages long to document the process in the event that they were challenged in an infringement suit. Phoenix also took sworn affidavits that its programmer had never seen the source code for IBM BIOS and even offered IBM a chance to examine their code. Phoenix had been so thorough that IBM has never challenged them in court.[676]

The "clean room" process was formalized during the first wave of copyright litigation as a means of developing interoperable software. The network effects driving software users and programmers made interoperability critical to competition in the software industry. Nearly every major litigation from *Apple v. Franklin* in 1983 through the *Oracle v. Google* litigation involves interoperability issues.

As illustrated in Phoenix Technologies' successful emulation of the IBM PC BIOS,[677] enterprises during the first API copyright wave faced two challenges in developing interoperable software: (1) discerning the functional specifications of the target computer program or system; and (2) independently developing interoperable software.[678]

The clean room process typically involves three teams of engineers and legal specialists. The first team — referred to as the "specification" or "dirty room" team — works with the target software to

676. *See* Russell Moy, *A Case Against Software Patents*, 17 SANTA CLARA COMPUTER & HIGH TECH. L.J. 67, 72–73 (2000) (footnotes and citations omitted).

677. *See Phoenix Technologies*, WIKIPEDIA (Sept. 8, 2017), https://en.wikipedia.org/wiki/Phoenix_Technologies [https://perma.cc/4T5A-9KNL] (discussing cloning of the IBM PC BIOS).

678. *See generally* P. Anthony Sammi, Christopher A. Lisy & Andrew Gish, *Good Clean Fun: Using Clean Room Procedures in Intellectual Property Litigation*, 25 INTELL. PROP. & TECH. L.J. 3 (2013); Jorge Contreras, Laura Handley & Terrence Yang, NEC v. Intel: *Breaking New Ground in the Law of Copyright*, 3 HARV. J.L. & TECH. 209 (1990); G. Gervaise Davis III, *Scope of Protection of Computer-Based Works: Reverse Engineering, Clean Rooms and Decompilation*, 370 COMPUTER L. INST., 115 (PLI Patents, Copyrights, Trademarks, and Literary Property Practice Course Handbook Series No. G-370, 1993).

determine the functional specifications.[679] It was initially unclear whether these programmers could make copies of target program code to decipher its functional specifications. Sega targeted that intermediate copying in its litigation strategy because the software that Accolade ultimately manufactured and distributed in its video games only copied unprotectable code elements required for interoperating with Sega's console. The Ninth Circuit's decision in the *Sega* case established that the intermediate copying by the "dirty room" team constituted fair use.

A second "coordination" or "audit" team, comprised of attorneys and engineers, establishes clear ground rules for managing the clean room process, screens programmers for the "clean room" team so as to ensure they have never seen the copyright-protected code, documents the activities and communication of the "dirty room" and "clean room" teams, oversees the process, and advises on what constitutes functional specifications and how to determine code segments that are unprotectable — segments that are unoriginal, standard programming practices, and necessary for interoperability or to accomplish specific processes or methods.[680] The coordination team seeks to ensure that no copyright-protected expression or misappropriated trade secrets get communicated to the clean room team.[681] It is only after those checks are completed that the process of independently coding an interoperable program commences.

The functional specifications detailing the particular processes or results that the target program accomplishes is then passed to the "clean room" team of programmers. This team remains shielded from the copyright-protected code. It designs, writes, and tests code aimed at accomplishing the target functional specifications.

In the second API copyright wave, the first stage of the process is not required because platform companies like Sun and Cisco publish

---

679. *See* Duncan M. Davidson, Reverse-Engineering of Software, *in* COMPUTER SOFTWARE 1989: PROTECTION AND MARKETING, 95–114 (1989).

680. *See* Marc Visnick, *Forensic Code Audits: Sometimes GPL Can Be A Four-Letter Word*, *in* OPEN SOURCE SOFTWARE, 2005: CRITICAL ISSUES IN TODAY'S CORPORATE ENVIRONMENT, at 354–55 (PLI Intellectual Property, Course Handbook Ser. No. G-846, 2005); Norm Alster, *New Profits from Patents*, FORTUNE, Apr. 25, 1988, at 185, 190; Steven Burke, *Court Support of 'Clean Room' Cloning May Legalize Intel '386 Chip Work-Alikes*, P.C. WEEK, February 27, 1989, at 63; Douglas K. Derwin, Licensing Software Created Under 'Clean Room' Conditions, *in* COMPUTER SOFTWARE 1989: PROTECTION AND MARKETING 439, 448–49 (1989).

681. Clean room processes are also increasingly used to ensure that open source code subject to viral licensing requirements — most notably, the GNU General Public License — does not infect software. *See* HEATHER MEEKER, OPEN SOURCE FOR BUSINESS: A PRACTICAL GUIDE TO OPEN SOURCE LICENSING, chs. 8–10, 12 (2015); David A. Wormser, *Open-Source Software: The Value of 'Free',* 22 INTELL. PROP. & TECH. L.J. 22 (2010); Ryan Paul, *Surveys Show Open Source Popularity on the Rise in Industry*, ARS TECHNICA (Jan. 20, 2006), http://arstechnica.com/uncategorized/2006/01/6017-2 [https://perma.cc/PM8J-U8US].

their interface specifications — API declarations — in an effort to encourage widespread learning and use of their programming language and platforms. Hence, Google could go directly to the second stage. But even that was costly and time consuming.[682]

Oracle challenged the authenticity of Google's clean room process during the first trial, but their evidence of cheating was weak.[683] Thus, since Google succeeded in its independent creation defense regarding the implementing code, the litigation should have ended without any need for a fair use defense.

b. Abstraction-Filtration-Comparison

Even if factual copying is conceded or proved, the copyright owner must further establish that the amount and significance of copying exceeds copyright law's infringement threshold. This standard is typically stated as substantial similarity of protected expression,[684] although some courts apply a higher "virtual identity" or "bodily appropriation" standard where the plaintiff's work is very simple and hence only narrowly protected.[685]

Even without using a formal clean room process, software developers often integrate pre-existing pieces of software code — both intentionally and unintentionally. Companies often hire programmers

---

682. *See supra* notes 322–26.

683. Oracle sought to show that a consulting firm (Noser Engineering) to which Google had outsourced some of the clean room effort had cut corners. *See* Closing Argument of Michael Jacobs, Trial Tr. at 207, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) ("The Google developers tried to claim that Android was developed in a clean room. But you know the clean room is dirty. There's copied code. There's decompiled code. You heard that the contractors that they used at Noser were super shady.") The principal source of evidence was an ambiguous Email communication. *See* Email from Bob Lee to Hiroshi Lockheiner and Dan Bornstein (Jan. 7, 2009), Trial Ex. 281, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. C 10-03561 WHA) (reproducing Jan. 6, 2009 Email from Hiroshi Lockheimer stating: "I'm a little nervous about signing Noser up to do any more work for us — but that's from a purely business perspective. Those guys (their management team) are super shady. (I know the engineers are great and I get that sense too from my limited interactions with them.)"). Oracle was unable, however, to prove direct copying of Java API source code beyond a few minor occurrences. *See Oracle v. Google*, 872 F. Supp. 2d at 982–83 (concluding that "Oracle has made much of nine lines of code that crept into both Android and Java. This circumstance is [] innocuous and overblown by Oracle . . .").

684. *See* NIMMER ON COPYRIGHT, *supra* note 96, at § 13.03[A].

685. *See* Harper House, Inc. v. Thomas Nelson, Inc., 889 F.2d 197 (9th Cir. 1989) (visual layout of a day planner comprised of a calendar and ruled lines); *see also* Mattel, Inc. v. MGA Entm't, Inc., 616 F.3d 904 (9th Cir. 2010) ("sculpt" for human-based dolls); Charles W. Ross Builder, Inc. v. Olsen Fine Home Bldg., LLC, 827 F. Supp. 2d 607, 611–12 (E.D. Va. 2011) (architectural works in which "nearly every design element of the two houses at issue" was dictated by external constraints for Colonial Williamsburg), *vacated*, 496 Fed. App'x. 314 (4th Cir. 2012); Incredible Techs., Inc. v. Virtual Techs., Inc., 400 F.3d 1007 (7th Cir. 2005) (visual elements of a golf video game); Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994) (largely unoriginal (and licensed) graphical office icons); Satava v. Lowry, 323 F.3d 805 (9th Cir. 2003) (a jellyfish sculpture encased in a domed glass cylinder).

from other companies who bring code with them.[686] Programmers often find useful code through Internet searches and integrate that code into their projects. In addition, programmers use standard coding techniques and logical structures that can produce similarities.

The *Altai* court developed a principled, systematic framework for determining copyright infringement of computer software code.[687] It adapted Judge Learned Hand's classic levels of abstraction/limiting doctrines framework[688] to the computer software medium. Applying this test often requires expert witnesses to explain computer programming design, languages, standard practices, and the various constraints.

The abstraction-filtration-comparison framework can come into play where the alleged infringer has used a clean room. Like a court deciding copyright infringement of software code, the coordination team of a clean room process must make calls about which elements of a target program are unprotected. As the *Altai* court noted, "[d]rawing the line between idea and expression is a tricky business."[689] The coordination team must apply various complex copyright doctrines, such as originality, merger, *scènes à faire*, and the § 102(b) exclusions. Thus, as in the *Altai* case, the plaintiff may challenge the legitimacy of the independent creation defense by alleging that protected literal code or non-functional SSO made it across the transom to the clean room and into the final product.

iii. Other Software Elements

Computer software can involve a variety of other potentially copyrightable elements. Many software programs include textual elements, including programmer comments, and audio-visual elements. Web sites feature expressive design elements and compilations of elements, some of which are functional. Most forms of conventional copyrightable works — books, pictorial and graphic images, music, motions pictures — are distributed through digital media and websites and hence are embedded within software. Graphical user interfaces, such as desktop icons, have compilations of pictorial elements.[690] Sculptural works can be represented as three-dimensional computer-aided design files.[691]

---

686. *See, e.g.*, *supra* text accompanying notes 87–88, 91–92, and 101–04.

687. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

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

689. *See Comput. Assocs.*, 982 F.2d at 704.

690. *See* Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.2d 1435 (9th Cir. 1994).

691. *See* Haritha Dasari, Note, *Assessing Copyright Protection and Infringement Issues Involved with 3D Printing and Scanning*, 41 AIPLA Q.J. 279 (2013).

Copyright applies to the literary, PGS, musical, and other works embedded in software to the same extent and in the same manner that it would apply to these works in analog form. The software encoding of the works may create derivative work to the extent that it translates, recasts, or transforms the preexisting underlying work.

The *Oracle v. Google* litigation highlights the pitfalls of a monolithic, one-size-fits-all approach to copyright protection of computer software. Many of the problems with the Federal Circuit's analysis trace to its treating declaring code as protectable expression as opposed to the functional specifications for a particular machine. By reimplementing that code in a clean room — that is without copying, but rather independently producing a code set that achieves the same functions as the Java APIs — Google steered clear of infringing copyright in the Java API components.

B. Policy Analysis

This Section assesses the extent to which the proper legal framework for copyright protection for computer software, as articulated in Section IV.A.3, comports with policy analysis. Section 1 presents an economic framework for analyzing legal protection for computer software. Section 2 traces the evolution of software markets. Section 3 explains that the proper legal framework for copyright protection for computer software best promotes software innovation and competition. Section 4 discusses the problems that the *Oracle v. Google* decision poses for achieving the proper copyright balance.

1. Economic Analysis of Legal Protection for Computer Software

Economic analysis of legal protection for computer software addresses two market failures: (1) the public problem associated with technological innovation; and (2) network effects associated with network technologies.[692]

i. The Public Goods Problem

Competitive markets generally provide an efficient allocation of resources when all of the costs and benefits of producing and distributing goods and services are reflected in market prices.[693] The government need only secure economic exchange through protection of private property and effective contract enforcement to promote economic efficiency. It is only when costs or benefits of producing and

692. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4.

693. *See generally* PAUL A. SAMUELSON & WILLIAM D. NORDHAUS, ECONOMICS (12th ed. 1985).

distributing goods are not fully reflected in market prices, such as when factories emit pollution without bearing the full social cost, that additional government intervention may be necessary to better promote economic efficiency.

Markets for many basic commodities, such as wheat, function relatively well without significant government intervention.[694] By contrast, markets for goods embodying intellectual work exhibit a significant market failure commonly referred to as the "public goods" problem.[695] Economists use the term "public goods" to describe goods that confer benefits upon the general public without individual members of the public having to pay for the goods. Classic examples include national defense and lighthouses. Public goods have two distinguishing features: (1) nonexcludability — it is difficult to exclude those who do not pay for the good from consuming it; and (2) nonrivalrous competition — additional consumers of the good do not deplete the quantity of the good available to others. A lighthouse warns all ships of rocky shoals whether or not they have paid for the service; and one ship's "enjoyment" of the lighthouse beacon does not diminish other ships' benefit of the warning. Although the private market can generate some investment in building warning beacons,[696] it is likely to be less than the optimal level.[697]

The research and development involved in technological innovation or the words of a novel falls within the public goods domain. Without some special form of legal protection, such intellectual work is nonexcludable — competitors can copy the innovation (a better mouse trap) or literary work (a HARRY POTTER novel) without bearing the costs of research and development. Furthermore, knowledge is nonrivalrous — an individual's enjoyment of the innovation or literary work does not reduce the availability of the good to others. Free markets will compete the price of the good to the marginal cost of production and distribution without taking into consideration the costs of research and development. As a result, the private market will tend

694. Even wheat markets, however, may require some government intervention. For example, there may be concerns about food safety.

695. *See* Peter S. Menell & Suzanne Scotchmer, Intellectual Property Law, *in* 2 HANDBOOK OF LAW AND ECONOMICS 1474 (A. Mitchell Polinsky & Steven Shavell eds., 2007). Even wheat markets may be affected by innovation. For example, scientists might be able to improve wheat crops through plant breeding.

696. *See* Ronald H. Coase, *The Lighthouse in Economics*, 17 J.L. & ECON. 357 (1974) (challenging the traditional view that lighthouses are examples of public goods by showing that privately owned lighthouses existed in England); *see also* William Barnett & Walter Block, *Coase and Van Zandt on Lighthouses*, 35 PUB. FIN. REV. 710 (2007) (contending that private lighthouses are possible and could obtain fees through voluntary clubs and turning off the light to free riders).

697. *See* Elodie Bertrand, *The Coasean Analysis of Lighthouse Financing: Myths and Realities*, 30 CAMBRIDGE J. ECON. 389 (2006); David E. Van Zandt, *The Lessons of the Lighthouse: 'Government' or 'Private' Provision of Goods*, 22 J. LEG. STUD. 47 (1993).

to undersupply these goods because producers cannot reap the marginal (incremental) value of their investment in providing such goods.[698]

The provision of intellectual property protection can address the public goods problem associated with innovation and expressive creativity. By affording exclusive rights to inventions and writings, the government enables inventors and authors to appropriate a return on the investments in research and development.

Such protection, however, imposes several social costs upon consumers, competitors, and other inventors. Intellectual property enables inventors and authors to charge more than the marginal cost of supplying goods, thereby raising prices and resulting in what economists refer to as "deadweight loss" — an equilibrium in which some consumers who are willing to pay more than the marginal cost of a product are priced out of the market.

Monopolistic exploitation distorts market pricing in the short run and can significantly affect entry and cumulative innovation over longer time horizons. Exclusive use of patented technology can inhibit cumulative inventors who seek to improve upon patented technology. Historical and industry studies of the innovation process find that inventions are highly interdependent: "Technologies . . . undergo a gradual, evolutionary development which is intimately bound up with the course of their diffusion."[699] In fact, "secondary inventions" — including essential design improvements, refinements, and adaptations to a variety of uses — are often as crucial to the generation of social benefits as the initial discovery.[700] Although licensing can alleviate this concern, it can also entail significant transaction costs.

Furthermore, the costs of determining the existence and scope of intellectual property rights can hinder competition and cumulative innovation.[701] In addition, administering intellectual property regimes and adjudicating disputes further contributes to social cost. Thus, the desirability and contours of intellectual property protection depend upon a balancing of social benefits and costs as well as consideration of alternative ways of promoting innovation and creativity.

Where the innovative aspect of a product can be hidden from public view — for example, through encryption or contractual restrictions — the innovators can appropriate a direct return for research

---

698. *See* Menell & Scotchmer, *supra* note 695, at 1477.

699. *See* Menell, *Challenges*, *supra* note 4, at 2645; *see generally* Nathan Rosenberg, *Factors Affecting the Diffusion of Technology*, 10 EXPLORATIONS ECON. HIST. 3 (1972).

700. *See, e.g.*, John L. Enos, *A Measure of the Rate of Technological Progress in the Petroleum Refining Industry*, 6 J. INDUS. ECON. 180, 189 (1958); James Mak & Gary M. Walton, *Steamboats and the Great Productivity Surge in River Transportation*, 32 J. ECON. HIST. 619, 625 (1972).

701. *See* Peter S. Menell & Michael J. Meurer, *Notice Failure and Notice Externalities*, 2013 J. LEG. ANAL. 1.

and development from direct consumers. Trade secret protection complements and reinforces this appropriation strategy. While secrecy will not work for book authors who must expose their creative product to the public for consumers to enjoy their works, secrecy has been successfully used in various sectors of the software where the source code can be hidden from view. For example, Google protects its core search technology by only returning search results to users.

Software producers can use technological means for preventing those who do not pay for the good from enjoying the benefits. For example, anti-copying devices can impede reproduction and disclosure of intellectual work embodied in products. World of Warcraft uses encryption to regulate access to its online multi-player platforms and charges consumers a subscription fee for access. That access can be disabled if the user violates platform rules or if their credit card does not process the subscription fee.

Beyond secrecy and technical protection measures, the first to introduce a product into the marketplace can in some contexts earn substantial and long-lived advantages in the market. Competitors in many product and service areas will need some time to ramp up their supply, thereby enabling first-movers to front-run competition and charge above marginal cost until the competition catches up and possibly longer. Moreover, innovative companies can develop a strong reputation for innovation, which can translate into loyal customers who are willing to pay premium prices. In this way, trademark protection reinforces the economic return of innovative first-movers.

First-movers can capitalize on their market-leading position by continuing to upgrade their products. This is a common strategy in the software marketplace. First-movers can also develop long-term relationships with customers, thereby ensuring a steady revenue flow. They can also deepen these relationships through service contracts and by customizing products and services for particular customers.

Companies can also subsidize their research and development by developing ancillary means of appropriation for research and development. Television and radio networks funded development of content by interleaving commercial advertisements. Such multi-sided market strategies have proven especially effective in many software industries. Sun Microsystems, for example, used Java to support its hardware business and licensing platform products that build on the Java language. Google successfully underwrote its search engine by developing a successful and innovative advertising business.

Government research and development subsidies have been extremely important in the development of computer technology. Moreover, universities, whose work product is often in the public domain, have played an important role in the development of computer technology.

The linkage between intellectual property protection and social welfare is greatly complicated in markets for products in which innovation occurs at many stages. What ultimately determines the social value of technological progress is the speed at and extent to which new, improved, and less expensive products are available. The number and type of individual technological innovations at particular intermediate stages are important, but no more important than the pattern of adoption and diffusion of these innovations.[702]

These interactions have been and continue to be particularly important in the evolution of computer technology. Advances in computer technology are made at many interrelated levels — basic research, hardware and devices, operating systems, program languages, APIs, application programming, network design, security — by diverse individuals, firms, and research institutions. It cannot be assumed automatically, therefore, that expansive legal protection for intellectual property at any one level will generate both the optimal amount of innovation and the optimal diffusion path.

ii. Network Externalities

The second principal market failure in the computer software market arises from the presence of network externalities. Network externalities exist in markets for products for which the utility or satisfaction that a consumer derives from the product increases with the number of other consumers of the product.[703] Telephone networks illustrate the phenomenon. The benefits to a person of a particular network depend on the number of other people connected to that same telephone network; the more people on the network, the more people each person can call and receive calls from. Robert Metcalfe, one of the inventors of Ethernet technology,[704] estimated that the value of a telecommunications network is proportional to the square of the number of connected users of the system.[705]

---

702. *See* Paul A. David, Technology Diffusion, Public Policy, and Industrial Competitiveness, *in* THE POSITIVE SUM STRATEGY: HARNESSING TECHNOLOGY FOR ECONOMIC GROWTH 373–92 (Ralph Landau and Nathan Rosenberg eds.,1986).

703. *See* Michael Katz & Carl Shapiro, *Network Externalities, Competition, and Compatibility*, 75 AM. ECON. REV. 424 (1985); DAVID HEMENWAY, INDUSTRYWIDE VOLUNTARY PRODUCT STANDARDS (1975).

704. Ethernet technology connects computers into a network. Its cables and software were standardized in the 1980s and became the foundation for many important advances in network technologies, including the Internet.

705. *See Metcalfe's Law*, WIKIPEDIA (Sept. 4, 2017), https://en.wikipedia.org/wiki/Metcalfe%27s_law [https://perma.cc/Z9HL-U9MW].

Another classic network externality flows from the prevalence of a standard typewriter keyboard.[706] Because almost all English language typewriters feature the same keyboard configuration, commonly referred to as "QWERTY," typists need learn only one keyboard system. Languages — from human to programming — also generate network effects. They enable communication, expression, and accomplishment of tasks.

Network externalities also inhere in product standards that allow for the interchangeability of complementary products and communication among devices.[707] Computer operating systems provide a compatibility nexus for interaction between the components of a computer system. The IBM BIOS, for example, established the protocols for using the IBM personal computer system. Sega programmed its Genesis video game console to run only those video game disks with a key for the lock-out code. Macros written for the Lotus 1-2-3 spreadsheet will only run on computer systems that recognize Lotus's menu command hierarchy. The declarations of the Java API packages enable particular API functionality. These codes and functional specifications control interoperability and particular functionality.

Consumers benefit when they and their devices, systems, and programs "speak" the most widely adopted platform — the *lingua franca* — or can translate that code into language that their devices understand. This often provides for greater functionality, such as more software that will run on their platform, and larger communication networks. At the same time, widely adopted product standards can strand the industry on an obsolete platform.[708] Consumers dislike the switching costs of learning new tools and languages. We often need strong reasons to jettison our well-worn devices and software tools for the less familiar. But occasionally, break-through products and software, such as versatile smartphones, can win us over.

Thus, reflecting durable good investments and human capital (training) specific to the old standard, the installed base built upon the dominant platform can create an inertia that makes it much more difficult for any single producer to break away from the old standard by introducing an incompatible product, even if the new standard offers significant technological improvement over the current standard.[709] In

---

706. *See* Paul A. David, *Clio and the Economics of QWERTY*, 75 AM. ECON. REV. 332 (1985) (in Vol. 75, No. 2, Papers and Proceedings of the 97th Annual Meeting of the American Economic Association).

707. *See* HEMENWAY, *supra* note 703.

708. *See* Joseph Farrell & Garth Saloner, *Standardization, Compatibility, and Innovation*, 16 RAND J. ECON. 70 (1985).

709. *See* HEMENWAY, *supra* note 703, at 30, 39; Joseph Farrell & Garth Saloner, *Installed Base and Compatibility: Innovation, Product Preannouncements, and Predation*, 76 AM. ECON. REV. 940, 940 (1986); Farrell & Saloner, *supra* note 708, at 71–72, 75–79.

this way, network externalities can retard innovation and slow or prevent adoption of improved product standards.[710]

Thus, companies seeking to leapfrog a widely adopted standard face substantial risk. They not only must invent an improved platform but they must also devise and execute a successful strategy to migrate consumers from the dominant platform. They also face the challenge of encouraging other software and complementary product developers to build for the new platform. One strategy is to steeply discount the costs of the new platform or to give access away for free. This is not a sustainable strategy unless the platform developer has ancillary revenue streams to cover their research, development, product, and support costs.

Another strategy is to build a convenient bridge over which consumers can easily migrate to and become accustomed to a new platform. Borland's motivation for building the Lotus 1-2-3 emulation mode was to support such migration. Similarly, Java's "Write Once, Run Anywhere" approach provided programmers with the ability to write web applications that could run on multiple hardware platforms.

A third strategy is to coordinate with other companies to jointly develop and market a new platform.[711] Standard setting organizations play an important role in overcoming bandwagon effects. They also facilitate cross-licensing and fair licensing of intellectual property rights.[712]

Intellectual property protection both contributes to and alleviates the network externality dilemma. On the one hand, intellectual property protection for the network features of computer technology can discourage realization of positive network externalities by limiting access to network technologies. The intellectual property owner can exclude competitors or charge a licensing fee for access, thereby raising costs. The intellectual property owner can also limit innovation. On the other hand, intellectual property protection can provide valuable incentives for overcoming bandwagon effects that entrench obsolete platforms.[713] Without the potential for a large reward, the platform innovator might not be able to surmount the technological and marketing challenging of leapfrogging the dominant platform.

---

710. *See* Farrell & Saloner, *supra* note 708, at 75–79.

711. *See* Timothy Simcoe, *Standard Setting Committees: Consensus Governance for Shared Technology Platforms*, 102 AM. ECON. REV. 305 (2012); Joseph Farrell & Timothy Simcoe, *Choosing the Rules for Consensus Standardization*, 43 RAND J. ECON. 235 (2012); Stanley Besen & Joseph Farrell, *Choosing How to Compete: Strategy and Tactics in Standardization*, 8 J. ECON. PERSP. 117 (1994); Joseph Farrell & Garth Saloner, *Coordination Through Committees and Markets*, 19 RAND J. ECON. 235 (1988).

712. *See* Contreras, *Patents, Technical Standards and Standard-Setting Organizations*, *supra* note 19; Contreras, *A Brief History of FRAND*, *supra* note 21; Lemley, *supra* note 19.

713. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4, at 1343.

As I explored in my early scholarship, the optimal design of intellectual property protection for addressing the network externality challenge is to protect the functional features of computer software under a limited utility patent-type regime, although with shorter duration and more flexibility to gain access to platforms that become widely adopted.[714] I advocated a genericide-type doctrine[715] which could protect emerging platforms but give way to broader access when a platform becomes dominant and risks affording the proprietor the ability to leverage that control to hinder cumulative innovators. This analysis anticipated Microsoft's rise and its abusive market tactics in undermining Netscape and Sun.[716] At the same time, I opposed copyright protection for the functional and interoperable aspects of computer technology to avoid large returns to first movers that win a standards battle without offering significant technological innovation and to afford competitors to use and build on unpatented methods of operation.

These ideas were referenced in the *Apple v. Microsoft* litigation,[717] where Apple sought to leverage its early lead in graphical user interface to control valuable, but largely unoriginal, interface features. The *Altai* court drew on analogous considerations in developing the abstraction-filtration-comparison framework for ensuring that copyright protection does not extend to the functional features of computer software.[718] Similarly, the *Lotus* court recognized the importance for competition and innovation of original developers and competitors being able to use and build upon unpatented methods of operation.[719]

The next Section explores how the evolution of software markets over the past two decades has added to our understanding of the proper role for intellectual property protection of network and functional features of computer software.

---

714. *See id.* at 1364–66.

715. *See* Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, *supra* note 4, at 1101. Under trademark law, a trademark can lose protection if it becomes associated in the public's mind with a category of product rather than the source of a particular brand of the product. *See, e.g.*, The Murphy Door Bed Co. v. Interior Sleep Sys., Inc., 874 F.2d 95 (2d Cir. 1989) ("Murphy bed" for a bed that folds up into a wall cabinet); Miller Brewing Co. v. Falstaff Brewing Corp., 655 F.2d 5 (1st Cir. 1981) ("Lite" beer); King-Seely Thermos Co. v. Aladdin Indus., Inc., 321 F.2d 577 (2d Cir. 1963) ("Thermos" vacuum insulated bottle).

716. *See supra* notes 227–48.

717. *See* Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1025 (N.D. Cal. 1992), *aff'd in part, rev'd in part*, 35 F.2d 1435 (9th Cir. 1994).

718. *See* Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 697–98, 705, 708, 712 (2d Cir. 1992).

719. *See* Lotus Dev. Corp. v. Borland Int'l, Inc.., 49 F.3d 807, 818 (1st Cir. 1995); *see also id.* at 819–21 (Boudin, J., concurring).

2. The Evolution of Software Markets

The first API copyright wave developed during a formative stage of the software industry during which the scope of copyright protection was uncertain, the programmer-driven open source movement was in its infancy, and the Internet had not yet emerged. Established companies, venture capitalists, and entrepreneurs looked primarily to proprietary strategies to appropriate a return on their investments in research and development. They saw strong legal, technological, and contractual protection for APIs — which control the network features of products — as critical to building their software kingdoms.[720] One entrepreneur from this era analogized creating an API to building a city:

> First you try to persuade applications programmers to come and build their businesses on [your tract of land]. This attracts users, who want to live there because of all the wonderful services and shops the programmers have built. This in turn causes more programmers to want to rent space for their businesses, to be near the customers. When this process gathers momentum, it's impossible to stop.
>
> Once your city is established, owning the API is like being the king of the city. The king gets to make the rules: collecting tolls for entering the city, setting the taxes that the programmers and users have to pay, and taking first dibs on any prime locations (by keeping some APIs confidential for personal use).[721]

Many of the first API copyright wave involved proprietary appropriation strategies — such as the use of lock-out codes. The Microsoft antitrust case litigated the extent to which a dominant player could leverage its intellectual property rights to control the Internet.

The rise of the Internet and advances in information storage and communication technologies in the late 1990s and early 2000 period significantly shifted software development strategies away from the proprietary model. Following the early success of Netscape's open strategy for distributing its Navigator browser,[722] Sun chose an even

720. *See* JERRY KAPLAN, STARTUP 49–50 (1995) (explaining that "'our value is the APIs"' and "[t]he real wars [in the computer industry] are over control of APIs" (quoting an industry remark)).

721. *Id.* at 150.

722. Microsoft would do Netscape one better by integrating its browser, Explorer, into its Windows operating system, making the marginal cost for Explorer essentially zero. This became one of the central issues in the government's antitrust enforcement action against

more permissive strategy for Java. Sun promoted widespread adoption through community outreach. It invited companies and programmers to collaborate in evolving the platform. Sun benefitted in a variety of ways from Java's rapid adoption and widespread use. Java's success modestly complemented its hardware business and maintained Sun's reputation and salience even as its core hardware business declined. More importantly, Sun blunted Microsoft's efforts to pull web development into its desktop monopoly.

In the wake of Sun's bold strategy of encouraging widespread use of Java through free and open access, other major platform developers followed similar strategies. Google successfully popularized its search engine by offering free search results. Its multi-sided market strategy — whereby advertisers would bid for keyword-triggered advertisements — revolutionized software economics. The power of network economics proved especially robust in the Internet age. Interoperability was only a part of the story. A variety of breakthrough software entrepreneurs found they could succeed by giving away software services in exchange for user data.

As the Internet took root as a prime economic driver for software and e-commerce, many companies (and their programmers) came to see open source software as an effective way of reducing software development costs and pre-committing to collaborative, non-predatory business practices. The open source movement transitioned from a quirky programmer protest movement to a mainstream business strategy.[723] Companies could ensure that their products and services would remain competitive, drawing in critical mass for establishing sustainable platforms. Open source infrastructure companies, such as Apache Software Foundation[724] and Mozilla,[725] displaced proprietary software providers. Even IBM, a staunch advocate for robust intellectual property protection, embraced open source as a competitive alternative to Microsoft's market power.[726]

Moreover, Google, Facebook, and other software companies successfully deployed advertising and other ancillary appropriability strategies to dominate some of the most important software markets. Network effects and multi-sided markets enabled these companies to

---

Microsoft. But in the end, Google did Microsoft one better by developing a better browser, giving it away, and using keywords to trigger advertisements.

723. *See* Robert P. Merges, *A New Dynamism in the Public Domain*, 183 U. CHI. L. REV. 183 (2004); Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source*, 50 J. INDUS. ECON. 197 (2002); Josh Lerner & Jean Tirole, *The Economics of Technology Sharing: Open Source and Beyond*, 19 J. ECON. PERSP. 99 (2005).

724. *See Apache Software Foundation*, WIKIPEDIA (June 18, 2017), https://en.wikipedia.org/wiki/Apache_Software_Foundation [https://perma.cc/K6G3-H7VY].

725. *See Mozilla*, WIKIPEDIA (June 18, 2017), https://en.wikipedia.org/wiki/Mozilla [https://perma.cc/5JW4-2RUX].

726. *See* Merges, *supra* note 723.

make "free" an attractive price[727] and "open" an attractive strategy[728] for software development. The more "free" services they provided, the more dominant their platforms became.

Unfortunately for Sun, it lacked a robust ancillary revenue stream to support the Java platform. Sun had used an open source philosophy to build its model. Although it had amassed some utility patents, the validity and scope of those patents were open to question, and Sun had not enforced them, raising questions about equitable defenses.

Google's Android project married various Internet business strategies to maintain and extend the success of its search and related services while ensuring a prominent role in the mobile Internet economy. Google combined a familiar programming environment with a free and open strategy to attract handset manufacturers, telcos, and app developers. It piggybacked on programmers' familiarity with the Java programming language and APIs. Its model motivated telcos, handset makers, and app developers to innovate and compete in the mobile field, enabling Android to blow past Sun, Microsoft, RIM, and Symbian and, eventually, to surpass Apple in the mobile platform marketplace. The telcos gained a competitive alternative to Apple's restrictive iOS platform. And the less restrictive open source license enabled handset makers and telcos to innovate and compete without losing control of their software innovations through GPL licensing. Google profited indirectly by ensuring that its ad-based services were central to the Android mobile platform and through new mobile-based opportunities.

This is not to suggest that the Internet has fully driven out proprietary software models. Many sectors, including relational databases, customer relationship management, social networks, and multi-player online video games, continue to rely upon proprietary software strategies. Further advances in bandwidth, speed, and storage capacity have pushed software business into Internet-based clouds.[729] Oracle has long used cloud-based business models to run its core relational database business. Like Google's search engine, a growing number of software businesses are able to protect their code through Software as a Service (SaaS) cloud platforms. Nonetheless, many of the sectors that rely upon interoperability and coordination have shifted to open and collaborative development models.

---

727. *See* CHRIS ANDERSON, FREE: THE FUTURE OF A RADICAL PRICE (2009).

728. *See* Tim O'Reilly, *Open Source Paradigm Shift*, O'REILLY (June 2004), http://archive.oreilly.com/pub/a/oreilly/tim/articles/paradigmshift_0504.html [https://perma.cc/62A4-2XTB]; ERIC S. RAYMOND, THE CATHEDRAL & THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY (2001).

729. *See* Lothar Determann & David Nimmer, *Software Copyright's Oracle from the Cloud*, 30 BERKELEY TECH. L.J. 161 (2015); Lothar Determann, *What Happens in the Cloud: Software as a Service and Copyrights*, 29 BERKELEY TECH. L.J. 1095 (2014).

To a significant extent, platform developers have partially addressed the appropriability problem associated with software development through the formation of collaborative clubs. [730] Compared to the early development of software markets in the 1980s and early 1990s, the contemporary and foreseeable future software marketplace reflects a zoned landscape featuring different governance structures. Much of web infrastructure is governed through standard setting bodies and open source communities run by nonprofit organizations, such as the Linux Foundation, the Apache Foundation, and Mozilla.org. These enterprises develop and update core software platforms through collaborative processes. The cloud provides software vendors with a variety of appropriability options, ranging from open to closed. The ability to maintain software in the cloud, through SaaS models, affords complete control over software through trade secrecy. Yet developers of complementary products, such as smartphones, have a need to make their platforms available to handset makers, telcos, and app developers.

The major "open" platforms largely operate as communities, whether or not they are sponsored by a single company or organization. Some rely on formal standards bodies; other use informal alliances. We have moved past the stage where one major player, such as IBM in the mainframe and early microcomputer era or Microsoft in the microcomputer and early Internet era,[731] can exercise dictatorial control over a widely adopted platform governing many inter-related products and services. We have instead seen the triumph of open, community-based platforms in critical Internet infrastructure markets.

The mobile phone marketplace illustrates this evolution. Early entrants — ranging from Sun's open Java ME platform to Microsoft's proprietary approach for feature phones — gave way to Apple's iOS and Google's Android platforms for smartphones. Apple provides a closed platform whereas Android provides a relatively open platform that evolves through a community process. Given the importance of widespread adoption of such technology, it is difficult to imagine a new entrant being able to make significant inroads without a relatively open, community-based process.

---

730. *See* Suzanne Scotchmer, Clubs, *in* THE NEW PALGRAVE DICTIONARY OF ECONOMICS (Steven N. Durlauf & Lawrence E. Blume eds., 2d ed. 2008) (discussing the economics of clubs, whereby individuals or groups share an activity so as to achieve common benefits).

731. *See* Thomas A. Piraino, *Identifying Monopolists' Illegal Conduct Under the Sherman Act*, 75 N.Y.U. L. REV. 809, 888–89 (2000) (quoting a Microsoft manager's internal e-mail stating that "to control the APIs is to control the industry").

3. The Optimality of Limited Copyright Protection for Computer Software

As Section IV.B.1 explained, intellectual property policy aims to address two inter-related market failures: (1) the public goods problem — enabling innovators to appropriate sufficient return on their investment in research and development; and (2) the network externalities problem — encouraging realization of network externalities while avoiding excess inertia. Intellectual property protection is not a panacea. It entails administrative and monopoly costs and can hamper cumulative creativity. Other government policies and private ordering (such as formal and informal standard setting) may be able to address aspects of the market failures more effectively.

Furthermore, the design of copyright protection for computer software depends critically on the larger intellectual property landscape. It was evident by the mid-1970s that computer software did not fit neatly within the traditional forms of legal protection for intellectual property.[732] By its inherent nature as written work intended to serve utilitarian purposes, software straddles the line between patent protection and copyright protection. The functional features of computer software and machines fall within the patent system's domain. The importance of interoperability and compatibility bring trademark protection into play. In addition, software can often be protected through trade secret law. Developers can hide their programming by only releasing object code versions to the public. They can also employ password-protections and contractual limitations. Furthermore, software can be entirely shielded from users through cloud services which provide users with only the results of software processes.

The potential anti-competitive effects of intellectual property protection also come into play. Intellectual property protection and antitrust law interact in complex ways. Antitrust law aims to promote free competition. It recognizes, however, that patent and copyright protection can promote dynamic competition. Major technological advances can improve product quality and drive down costs. For example, Intel Corporation, relying on patent protection, has been able to enhance the performance of microprocessor chips by a factor of 3,500 while improving energy efficiency by a factor of 90,000 and reducing cost by a factor 60,000 over the past fifty years.[733] Because of its inherent

732. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4, at 1329 (citing CONTU REPORT, *supra* note 47, at 3).

733. *See* Thomas Friedman, *Moore's Law Turns 50*, N.Y. TIMES, May 13, 2015, at A27; *Moore's law*, WIKIPEDIA (June 18, 2017), https://en.wikipedia.org/wiki/Moore%27s_law [https://perma.cc/7BRH-2UQ6]; Gordon E. Moore, *Cramming More Components onto Integrated Circuits*, 38 ELECTRONICS 114 (1965) (predicting that the number of transistors in an integrated circuit would double approximately every two years).

limitations — barring protection for ideas, processes, systems, methods of operation, and discoveries — copyright protection, properly interpreted, posed little threat to competition.

Yet with intellectual property protection comes the potential for abuse. As the Microsoft antitrust litigation illustrated, network industries are especially prone to leveraging market power in one software field to hamper innovation and competition in other sectors. The antitrust doctrines seeking to resolve this tension are difficult to apply, especially in network industries, which are naturally prone to high concentration.

When I first wrote about legal protection for computer software three decades ago, the economic theory was relatively clear but the technological and market contexts were evolving rapidly. The experience of the past several decades have reinforced the insights of that earlier research. Although copyright law has a valuable role to play in protecting computer software, that role must be limited, especially with regard to network and other functional features of computer software.

As explicated in Section IV.A.3, the proper *legal* contours of copyright for protection for computer software — based on the seminal *Baker v. Selden* decision, the 1976 Act, the CONTU REPORT, and the nature of computer technology — distinguish between the functional specifications and the implementing code. The functional specifications fall outside of copyright protection regardless of their "creativity" and the difficulty of designing and coding them. Thus, the declarations of an API are unprotectable (under copyright law) as they are necessary to build a particular machine. Such features might be eligible for utility patent protection, but they would have to meet patent law's higher threshold requirements and the duration of such protection would be shorter than copyright protection.[734] By contrast, the implementing code is protectable, although particular code, such as bits required for interoperability, and particular design elements, such as standard programming techniques, might be filtered out as functional or unoriginal. Furthermore, copyright's fair use doctrine authorizes competitors to reverse engineer protected code to determine the unprotected elements.

---

734. Patent law has long provided protection for innovative interfaces. For example, Samuel F.B. Morse patented not only the machinery for telegraphic communication, but also the system of dots and dashes that came to be known as Morse Code. *See* O'Reilly v. Morse, 56 U.S. 62, 86 (1853) ("Fifth. I claim, as my invention, the system of signs, consisting of dots and spaces, and of dots, spaces, and horizontal lines, for numerals, letters, words, or sentences, substantially as herein set forth and illustrated, for telegraphic purposes."). Under *Baker v. Selden*, such a system is ineligible for copyright protection, even though a book explaining its use would garner thin copyright protection. Others could not reproduce the book, although they could write their own guide to the use of Morse Code.

This regime provides an effective tool for combating unauthorized distribution of software programs while affording competitors and other innovators freedom to use functional features and to develop interoperable products. At the same time, it ensures that utility patent law serves as the principal means for protecting functional software elements. This combination provides balanced incentives for promoting progress in software platforms while supporting the realization of network externalities. In the absence of patent protections for interface specifications, competitors can emulate the interface specifications in developing interoperable products or adapting platforms. Companies seeking to establish proprietary platforms can look to utility patent law.

Even though copyright law does not directly protect functional features of computer software, the thin layer of copyright protection for implementing code provides developers with valuable lead-time. By employing technological protection measures or distributing software products solely in object code form, software developers can slow development of interoperable products. Reverse engineering computer programs can be time-consuming and expensive.[735] Even when the functional specifications are publicly disclosed so as to rapidly expand the platform, as was the case with the Java APIs, reimplementing that functionality in a clean room can be time-consuming. It is often more efficient to write a program from scratch. But because of interoperability concerns or user and programmer familiarity with a software product, reverse engineering and reimplementing the precise functionality can be necessary to introduce a new or complementary product. Thus, copyright protection in conjunction with trade secret protection provides software developers with a first-mover advantage.

This interpretation of copyright protection for computer software provides a sound regime for promoting software innovation and competition. Thus, the software industry today can be analogized to a zoning map. It comprises distinct sectors with varying approaches for addressing the appropriability and network externality concerns.

The core Internet technologies and other high-level platforms develop almost exclusively through open source projects and standard setting processes. Many different constituencies — ranging from major corporations to governmental and non-governmental organiza-

735. *See* Sammi, Lisy & Gish, *supra* note 56, at 10 (discussing the high costs and risks associated with reverse engineering); Contreras, Handley & Yang, *supra* note 678, at 214 (same); Davis III, *supra* note 678, at 151 (noting that "it would be easier and far less expensive to develop entirely new software, were it not for the need in most such cases to have a functional equivalent, compatible program that cannot be obtained in any other way"); Burke, *supra* note 680, at 63 (noting that reverse engineering the IBM BIOS doubled Phoenix's cost of developing a functioning BIOS).

tions — have come to see that such platforms are too important to be owned by any one enterprise. Moreover, the open source community has proven especially effective at generating innovative research and development through collaborative processes that do not rely on corporate ownership or direct remuneration. Thus, this critically important area of software development has solved the public goods/network externalities problem without substantial reliance on exclusive intellectual property rights. Affording copyright protection to API design for this sector is neither needed nor desirable.

For entrepreneurs and companies operating within these high-level platforms, the Internet itself has largely solved the appropriability problem. The ability to operate software services in the cloud obviates distribution of software products. The proprietors of cloud services can largely protect their software through trade secret law, technological protection measures, and contractual provisions. Google's search engine, Facebook's social network, and countless other software-based companies secure much of their software without copyright protection.

Companies that choose to distribute their software products have clear protection against piracy of their software. They also can garner lead-time through technological protection measures and not distributing their source code. They cannot, however, protect the functional features of their product beyond the period necessary to reverse engineer and re-implement the uncopyrightable functional features. This comports with a proper channeling of protection between utility patent and copyright law.

We are left with the question of whether the lack of direct *copyright* protection for API design — whose interface must be exposed to the public in most commercial circumstances to be effective — creates an undesirable lacuna in intellectual property protection. Are incentives to innovate platforms inadequate without copyright protection for API design?

Utility patent law provides protection for novel, non-obvious, and adequately disclosed advances in computer systems, processes, and interface design. It arguably overprotects interface specifications for an excessive duration.[736] Thus, adding robust copyright protection for API design would further undermine realization of network externalities and hamper cumulative innovation. In the past three decades, the software industry has demonstrated that API design projects typically require large community-building efforts to overcome the inertia of widely adopted standards. Whereas pre-Internet enterprises looked to proprietary models to justify the research and development effort

736. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4, at 1364–65.

needed to surmount this inertia, modern software markets demand more open, collaborative approaches. As with core Internet technologies, this approach has largely surmounted the public goods/network externalities challenge. In this sector, ancillary appropriability means, such as advertising, have proven especially important.

API design innovation solves the network effects problem through community organization, formal and informal standard setting processes, and open licensing. It is nearly impossible to compete or supplant a widely adopted platform without the target audience buying in. Successful ventures are able to pair such community organization with indirect appropriability strategies. Leaving API design specifications outside of copyright protection enables entrepreneurs seeking to improve on successful platforms to build bridges for users and programmers. This avoids excess inertia and accommodates creative destruction and evolution[737] in those areas where the proprietor of the standard platform lacks patent protection.

Thus, looking back over the past three decades, the need for copyright protection to address the dual public goods/network externality problem face by software developers has substantially waned due to several factors. The emergence and development of the Internet has enabled software developers to distribute software and services at very low cost. Furthermore, developers can protect their code through cloud service models. The Internet has also opened up and expanded the effectiveness of e-commerce and advertising-based business models. More robust copyright protection for API design would likely have stifled platform innovation and competition. Thus, a parsimonious approach to copyright protection of computer software remains the best policy choice.

The evolution of the mobile platform illustrates the wisdom of excluding API specifications from copyright protection while affording the API implementing code limited protection. Sun built the Java programming language and API platforms on the C programming language. It successfully promoted Java through free and open licensing as well as through its establishment of the Java Community Process.[738] This strategy enabled Sun to thwart Microsoft's effort to monopolize web programming.

By the early 2000 period, Sun was well-positioned to lead the shift to mobile technology. Its Micro Edition gained a strong position in the feature phone marketplace. It failed, however, to recognize the potential for more versatile mobile devices. To some extent, it was

737. *See* JOSEPH A. SCHUMPETER, CAPITALISM, SOCIALISM AND DEMOCRACY 82–83 (1942) (describing economic evolution as driven by the "gale of creative destruction," a "process of industrial mutation that incessantly revolutionizes the economic structure from within, incessantly destroying the old one, incessantly creating a new one").

738. *See supra* notes 250–53.

held back by its focus on backward compatibility with the WORA principle and its restrictive licensing philosophy.

Google was able to leverage its highly profitable advertising business to support the development of Android — a mobile platform that provided full browser functionality as well as other mobile functionality. The Android team recognized that this operating system would need to fit on the small chips in handsets and accommodate some other capabilities, such as users' locations and preferences. They also saw the advantages of a more permissive licensing model.

Rather than build the platform entirely from scratch, the Android team sought to use the well-known Java programming language and some of the Java API packages. The Java programming language was freely available. The APIs, however, were protected by copyright law. Hence, Android developers reached out to Sun to negotiate a license. While desperate to generate more licensing revenue for the Java unit, Sun was reluctant to license a platform that did not have the full range of APIs necessary to extend "Write Once, Run Anywhere" interoperability. In addition, Sun opposed the more permissive license that Google sought to afford Android licensees.

Although Sun and Google came close to agreement in the spring of 2006, the negotiations reached an impasse. In order to avoid infringement of Sun's implementing code, Google undertook the costly and time-consuming process of re-implementing thirty-seven of the Java API packages in a clean room. The Google vision succeeded for many of the reasons that Java succeeded. Android enabled the target audience — handset manufacturers, telcos, and app developers — to quickly learn the platform and to compete. The Android platform provided a viable alternative to Apple's iOS platform.

This resulted in robust innovation and competition. Apple was not able to dominate the mobile platform in the manner that Microsoft monopolized the desktop. Google's power is checked in part by the need to work within the open handset alliance. Moreover, Google ceded substantial power to its partners. The smartphone ecosystem has been remarkably dynamic and competitive.

The critical question is whether the lack of strong copyright protection for API design stands in the way of the next great software platforms. The experience of the past several decades suggest that strong copyright protection for API design would more likely hinder rather than promote technological progress. Had Sun been able to stand in Google's way based solely on API design, we would never have gotten a bold new platform and the permissive licensing structure that has ignited competition and innovation among downstream handset manufacturers, telcos, and app developers. Apple's iOS would likely have dominated the mobile platform for a long time.

A subsidiary question is whether Android's forking of Sun's Java platform into an implementation that is not compatible with Java's WORA principle undermines network externalities. Oracle contends that copyright protection for API functional specifications is critical to enforcement of the GPL.[739] The Free Software Foundation ("FSF"),[740] which established and maintains the GPL, disagrees. In its amicus brief opposing Google's petition for certiorari following the Federal Circuit's 2014 decision, the FSF stated that it "strongly rejects the use of copyright law to prevent implementation of interoperable free software by inappropriately applying copyright principles to ideas instantiated in the rules of inter-program communication called 'application program interfaces.'"[741]

The fact that Android does not afford complete end user compatibility with Java does not necessarily lead to the conclusion that consumer or programmers will be harmed. Even Sun's Java Micro Edition was not fully interoperable with the Java Standard Edition.[742] More importantly, interoperability is but one of many functional considerations. Although complete interoperability can be an important programming goal, it can also stand in the way of technological progress. Google sought to draw on the Java APIs as part of its plan to develop a more versatile and compact platform optimized for a new generation of mobile devices. Woodenly adhering to the WORA principle would have compromised these important design objectives and forced desirable cumulative innovation onto a risky path. Forking of code is an essential part of creative destruction.

Furthermore, Google did not seek, as Microsoft did in the late 1990s, to undermine the Java platform in violation of a contractual agreement.[743] Nor did Google claim that Android was compatible with Java. Rather, it sought to implement particular function packages in a new, familiar, and partially interoperable mobile platform. Using some of the Java APIs provided a bridge for the millions of Java programmers. But by independently implementing the packages, Google sought to work around copyright protection in the implementing code. It avoided trademark liability by not using the Java trademark in a manner that created a likelihood of confusion. Google risked patent infringement, but ultimately fended off Oracle's patent assertions.

---

739. *See* Hurst, *supra* note 428.

740. *See Free Software Foundation*, WIKIPEDIA (June 18, 2017), https://en.wikipedia.org/wiki/Free_Software_Foundation [https://perma.cc/YXJ4-E843].

741. *See* Brief of Software Freedom Law Center and Free Software Foundation as Amici Curiae Supporting Respondent, Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015), *memorandum op.*, (No. 14-410) (2014 WL 6967821). Somewhat confusingly, FSF supported Oracle in opposing the Supreme Court granting certiorari, although on entirely different grounds.

742. *See supra* notes 505–06.

743. *See supra* text accompanying notes 236–48.

More importantly, protecting API design through copyright law poses a much greater risk to interoperability than the parsimonious approach that predated the Federal Circuit's *Oracle v. Google* decision. The many companies that build products that connect with established platforms would risk copyright infringement by building interoperable features. If they did not obtain licenses, they would potentially have to prove that their independently developed interface code was fair use.[744]

4. Impediments to Achieving the Proper Copyright Balance Posed by the *Oracle v. Google* Litigation

This analysis shows that the fair use trial was a massive waste of time, party resources, and judicial resources. The litigation has already established that Google independently implemented the functional specifications for thirty-seven "machines." As a result, it cannot be held liable for copyright infringement.

More importantly, even if Google ultimately prevails in this litigation under the fair use doctrine, the Federal Circuit's holding that API functional specifications are copyrightable will hamper software innovation and competition. Its 2014 decision revives long dormant fears about the scope of copyright protection for computer software.

The Federal Circuit has opened a Pandora's Box that will lead legal advisors to caution against independently implementing APIs without obtaining licenses. They will need to advise their clients that so long as there is the potential for a patent infringement allegation — which is likely in view of the proliferation and availability of software patents — then a company seeking to control access to its platform can likely file a lawsuit outside of the First Circuit alleging both patent and copyright infringement that would fall within the Federal Circuit's exclusive appellate jurisdiction.[745] As such, the district court would be on notice that the appellate court considers the functional specifications of APIs to be copyrightable. The defendant's principal hope will be to mount a fair use defense, which is notoriously unpredictable and costly. The range of factors applicable to fair use opens up a broad range of discovery.

The fair use doctrine is an especially poor vehicle for resolving API copyright disputes. As Judge Boudin recognized in his thoughtful concurrence in *Lotus v. Borland*, a "privileged use" doctrine, akin to

744. *See* Julie Samuels, Oracle v. Google *and the Dangerous Implications of Treating APIs as Copyrightable*, ELEC. FRONTIER FOUND. (May 7, 2012), https://www.eff.org/deeplinks/2012/05/oracle-v-google-and-dangerous-implications-treating-apis-copyrightable [https://perma.cc/D65R-G44D].

745. I exclude the First Circuit because the Federal Circuit would be bound by the First Circuit's *Lotus v. Borland* decision.

the fair use doctrine that he considered as an alternative to the "method of operation" exclusion, "would cause cost and delay, and would also reduce the ability of the industry to predict outcomes."[746]

Furthermore, it is possible that other circuits could follow the Federal Circuit. The Third Circuit likely already does.[747] The unusual posture of the *Oracle v. Google* litigation creates the risk that the Federal Circuit's copyrightability decision will go without review.[748] The Supreme Court has already declined one opportunity to review the matter.

Software developers and investors greatly value clarity in making the difficult, time-sensitive decisions involved in designing products and platforms. Yet the fair use defense is fact-dependent and case-specific. It is a classic example of a legal standard for which dispute resolution is costly and time-consuming.[749] The Android team struggled with these issues and ultimately chose what it thought would be a safe harbor: re-implementing the code. While the fair use trial partially vindicated its decision, the uncertainty imposed tremendous cost and ultimately took nearly a decade for resolution. And even that resolution remains in question as the appeal looms. For a fast-moving industry like software, this regime for determining freedom to operate is highly inefficient. It distorts and slows innovation activities.

Judge Boudin concluded his concurrence with this sage summary and observation:

> [T]he majority's result [based on § 102(b)'s exclusion of methods of operation from the scope of copyrightable subject matter] persuades me and its formulation is as good, if not better, than any other that occurs to me now as within the reach of courts. Some solutions (e.g., a very short copyright period for menus) are not options at all for courts but might be for Congress. In all events, the choices are important ones of policy, not linguistics, and they should be made with the underlying considerations in view.[750]

I would add only that *Baker v. Selden* and full consideration of the legislative history of the Copyright Act of 1976 support the First

---

746. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 821–22 (1st Cir. 1995) (Boudin, J., concurring), *aff'd by an equally divided court*, 516 U.S. 233 (1996) (per curiam).

747. *See* Section IV(A)(2)(ii)(c).

748. *See supra* Section III(C).

749. *See* Louis Kaplow, *Rules Versus Standards: An Economic Analysis*, 42 DUKE L.J. 557, 563 (1992).

750. 49 F.3d at 822.

Circuit's approach. That approach properly immunizes Android's use of the functional specifications for Java APIs, although Google could nonetheless infringe by failing to re-implement those specifications independently.

V. CONCLUSION

As we know from popular books and films, zombies (or vampires for pre-Millennials) tend to awaken at inopportune times and wreak havoc. Eventually some suitable leading actor re-inters the zombies or impales the vampire through the heart. We are not there yet. The Federal Circuit's flawed analysis brings the *Apple v. Franklin/Whelan* API copyright zombies back to life. The district court's fair use decision suggests that it will be difficult for copyrightable API zombies to proliferate, but it does not resolve the issues beyond the particular case and thus is of little to no precedential value. Furthermore, Oracle has appealed the jury's fair use ruling. Thus, API copyright zombies still roam Silicon Valley and other innovation centers.

This Article chronicles the API copyright battles. It shows that after some early missteps, the courts surmounted the copyright challenges posed by network and functional features of computer software by the mid 1990s. But the *Oracle v. Google* litigation threatens to set copyright law back to the misguided analyses of the 1980s. Part IV explains that copyright law's fundamental exclusion of protection for functional features dictates that the labeling conventions and packaging of functions within interface specifications generally fall outside of the scope of copyright protection even as implementing code garners thin copyright protection. This interpretation of copyright law comports with fundamental principles channeling protection among the modes of intellectual property protection. It also serves the larger goals of intellectual property law and competition policy.

Zombies thrive on the flesh and lifeblood of the living and are difficult to subdue. In the software community, confusion over copyright treatment of APIs has exposed developers and investors to chilling uncertainty that restrains and distorts the creative destruction that is critical to advancing computer technology and software industries. This Article has explained computer programming, legislative and jurisprudential history, the evolution of the software industry, and economic analysis in an effort to return the API copyright zombies to their rightful and hopefully final resting place.[751]

751. Software innovation and progress can also be advanced through patent reforms. I continue to believe that a *sui generis* regime would be the best approach. *See* Menell, *Tailoring Legal Protection for Computer Software*, *supra* note 4, at 1371. Nonetheless, reinterring the API copyright zombies is an important step that can be accomplished without legislation.

APPENDIX A: GLOSSARY

American Committee for Interoperable Systems ("ACIS"): A group of technology companies formed ACIS in the early 1990s to advocate for less protectionist intellectual property policies for computer software. As the name emphasized, ACIS focused on promoting interoperability. Sun Microsystems and Oracle were among ACIS's founding members.

Android SDK: The Android Software Development Kit is a comprehensive set of development tools for creating programs that will run on the Android platform.

Apache Harmony: Announced in May 2005, the Apache Harmony Project sought to develop an open source, free Java implementation. It sought a license from Sun, with mixed success. The Android operating system used this platform. The Apache Harmony project was retired in late 2011.

Application Programming Interface (API): An API is a set of subroutine definitions, protocols, and tools for building application software that functions as a clearly defined method of communication between various software components. APIs can be used for web-based systems, operating systems, database systems, computer hardware, or software libraries. (Wikipedia)

Berkeley Software Distribution (BSD): Berkeley Software Distribution is a UNIX operating system produced by the Computer Research Group at the University of California, Berkeley from 1977 to 1995. (Wikipedia)

Clean Room: A clean room process insulates programmers from copyright-protected code in producing code that accomplishes the same functions as a target program based solely on the functional specifications. Such a process ensures a program is independently written and hence not copied except with regard to unprotectable elements.

Code Forking: Forking of software code refers to the creation of an independent branch of a computer program. This split from the original program typically spawns competing projects that are not interoperable, resulting in a split in the software developer community.

CONTU: Congress established the National Commission on New Technological Uses of Copyrighted Works ("CONTU") in 1974 to study the implications of new technologies and recommend revisions to federal intellectual property law. Its 1978 Report recommended statutory changes to accommodate computer software within copyright law, which Congress implemented in 1980. The 1978 Report is treated by courts as legislative history and guidance for interpreting copyright protection for computer software.

Dalvik: Google's Android team developed Dalvik as an independent Java Virtual Machine for use with the Android platform. Dalvik was created using a clean room process.

Declarations/Declaring Code: "In computer programming, a declaration is a language construct that specifies properties of an identifier: it declares what a word (identifier) 'means.' Declarations are most commonly used for functions, variables, constants, and classes, but can also be used for other entities such as enumerations and type definitions. Beyond the name (the identifier itself) and the kind of entity (function, variable, etc.), declarations typically specify the data type (for variables and constants), or the type signature (for functions); types may also include dimensions, such as for arrays. A declaration is used to announce the existence of the entity to the compiler; this is important in those strongly typed languages that require functions, variables, and constants, and their types to be specified with a declaration before use, and is used in forward declaration." https://en.wikipedia.org/wiki/Declaration_(computer_programming)

DMCA: The Digital Millennium Copyright Act of 1998 amends the Copyright Act to address key challenges introduced by the Internet among other issues. The key Internet-related provisions are prohibitions on circumvention of technological protection measures and the establishment of online service provider safe harbors. The DMCA exempts decryption of technological protection measures for purposes of developing interoperable systems.

Free Software Foundation (FSF): The Free Software Foundation, established by Richard Stallman, seeks to promote users' rights to use, study, copy, modify, and redistribute computer programs.

Forking: See Code Forking.

GNU: "GNU's Not Unix!" is an operating and extensive collection of computer software developed by the FSF. GNU is composed wholly

of free software, most of which is licensed under the GNU Project's General Public License. (Wikipedia)

Green Project/Team: Code name for the Sun Microsystem team in charge of inventing Java, originally called Oak, developed starting in 1990.

General Public License (GPL): The Free Software Foundation establishes the General Public License, sometimes referred to as a "copyleft" license, to prevent programmers from building proprietary limitations into "free" software. GPL guarantees end users the freedoms to run, study, share (copy), and modify the software so long as the users permit use of any derivative works on the same terms.

Interoperability: Interoperability refers to the capability of hardware devices or software programs to interact. Interaction can occur from hardware devices to hardware devices (e.g., a printer connecting to a computer), from software to software (e.g., an application program running on a particular operating system, two application programs exchanging data), and software to hardware (e.g., a video game running on a game console, an app running on a particular mobile device). Interoperability can refer to the capability to communicate, execute programs, or transfer data among functional components of computer systems. Interoperability can be a question of degree, ranging from full interoperability in which all features of a system work seamlessly to partial interoperability in which some of features function together.

Java: Sun Microsystems developed the Java programming language and API platform for use in webpage programming. It implements a "Write Once, Run Anywhere" ("WORA") principle to facilitate ease of use of web pages across the full range of computer systems. Java is an object-oriented language.

Java Community Process ("JCP"): Sun Microsystems established the Java Community Process as a formal process for developing standard technical specifications for Java technology. The JCP is open to the public, including organizations and commercial entities.

Java Micro Edition ("Java ME"): A Java platform designed for embedded and mobile devices.

Java Standard Edition ("Java SE"): A Java platform designed for desktop or laptop computers.

Java Standard Edition API: Sun Microsystems organized sets of pre-written programs (methods, which are grouped in classes) into API packages (or class libraries). Each API package reflects a set of declarations or functional specifications needed to invoke the functions. It is executed through detailed implementing code. Although a Java programmer can write new code (methods) from scratch, the pre-written methods within the Java API packages provide convenient, efficient, reliable, standardized building blocks, thereby saving Java programmers tremendous tedious effort.

Java Specification Requests ("JSRs"): Java community members may propose Java Specification Requests for expanding and updating the Java platform. These are reviewed by the JCP through a public process akin to administrative rulemaking. The JCP Executive Committee, comprised of major stakeholders, decides whether to approve JSRs.

Java Virtual Machine ("JVM"): A Java Virtual Machine is an abstract computing machine that enables a computer to run Java programs.

Linux: A UNIX-compatible kernel developed by Linus Torvalds in 1991, licensed on the GNU GPL "open source" model.

Object Code: Computers manipulate data according to a set of instructions called a computer program. At their most basic level, computer programs represent information and instruct computer devices through binary information ("0" (usually connoting "off") and "1" (usually connoting "on")). Strings of binary information can represent alphanumerical symbols, words, and images. Programs written in high level, human-readable computer languages ("source code") must be compiled into computer-readable "object code." Object code is typically platform-specific — i.e., adapted to a particular computing system.

Object-Oriented Programming ("OOP"): The OOP paradigm structures software development around "objects," containing data and methods, that interact with each other. An object's procedures can access and modify data fields with which they are associated. This paradigm is more efficient and more easily modifiable and maintainable than more conventional procedural programming languages like C.

Original Equipment Manufacturer ("OEM"): The computer and telecommunications industries make extensive use of OEMs to supply

components, parts, and equipment marketed by other companies. (Wikipedia)

Source Code: Computer programs are typically written in high level, human-readable languages such as Fortran, C, and Java. Such "source code" programs are compiled using particular lexical, syntactic, and semantic rules into computer-readable "object code" for execution on a particular computer operating system. Source code is platform-independent and thus can be compiled for different computer operating systems.

Technology Compatibility Kit ("TCK"): A Technology Compatibility Kit is a suite of tests that checks an implementation of a Java Specification Request (JSR) for compliance with a Java Platform. (Wikipedia)

Telcos: Telecommunication companies.

Write Once, Run Anywhere ("WORA"): Sun Microsystems coined the term "Write Once, Run Anywhere" to characterize the interoperable software design philosophy underlying the Java language and platform. Programs written in Java can run on any chip, device, or software package equipped with a Java virtual machine. The Java TCK ensures that a product complies with Java compatibility.

APPENDIX B: PRINCIPAL PARTICIPANTS

Android: Software company founded in October 2003, acquired by Google in July 2005.

- **Andy Rubin**, former Apple engineer, founder of Android and later head of Google's Android division

Apple: Technology company founded in April 1976.

- **Steve Jobs**, founder and late CEO
- Apple's introduction of the iPhone in 2007 transformed the smart phone marketplace. Google played a role in developing iPhone applications. Apple's patent, design patent, and trade dress litigation complicated Google's efforts to establish the Android platform.

Federal Judiciary

- **Judge William H. Alsup**, judge assigned to *Oracle v. Google*'s district court trials
- **U.S. Court of Appeals for the Federal Circuit**, appellate court reviewing *Oracle v. Google* trial court decisions.

Google: Technology company founded in September 1998, which went public in 2004.

- **Sergey Brin**, co-founder of Google and current president of parent company Alphabet
- **Larry Page**, co-founder of Google and current CEO of parent company Alphabet
- **Eric Schmidt**, former CTO of Sun Microsystems, then CEO of Google from 2001–2011, currently executive chairman of Alphabet. Served on Apple's Board of Directors from 2006 to 2009
- **Robert Van Nest**, lead counsel in *Oracle v. Google* trials

Microsoft: Technology company founded in April 1975 by Bill Gates and Paul Allen. Microsoft's efforts to leverage its successful Windows platform to control the Internet browser technology and web programming pushed Sun toward an open development strategy for the Java language and platforms. Sun successfully sued Microsoft for breach of its Java license agreement and antitrust violations.

Oracle Corporation: Technology company founded in June 1977, which acquired Sun Microsystems in January 2010.

- **Larry Ellison**, co-founder, former CEO, and current Executive Chairman and CTO since September 2014
- **Safra Catz**, current co-CEO
- **Peter Bicks**, co-lead counsel in *Oracle v. Google* trials
- **Annette Hurst**, co-lead counsel in *Oracle v. Google* trials

Sun Microsystems: Technology company founded in February 1982, acquired by Oracle in January 2010. Created networked computer engineering workstations and the Java (previously Oak) programming language.

- **Vinod Khosla**, co-founder
- **Andy Bechtolsheim**, co-founder
- **Scott McNealy**, co-founder
- **Bill Joy**, co-founder, participant in Berkeley Software Distribution creation
- **Scott McNealy**, CEO until 2006
- **Jonathan Schwartz**, CEO from 2006 to 2010 (acquisition by Oracle)
- **Eric Schmidt**, CTO, later CEO of Google in 2001.
- **Tim Lindholm**, former Sun Microsystems engineer who was involved with Java. His Emails to **Andy Rubin** about Java were important exhibits in the trial.

Trial Witnesses for Google

- **Eric Schmidt**, Google CEO
- **Jonathan Schwartz**, former Sun Microsystems CEO (at time that Android was developed)
- **Andy Rubin**, head of Android team
- **Larry Ellison**, Oracle co-founder
- **Joshua Bloch**, former Sun employee who became Google's "Java guru"
- **Donald Smith**, designated Oracle representative
- **Simon Phipps**, Sun Microsystem's former Chief Open Source Officer and former President of the Open Source Initiative
- **Daniel Bornstein**, key member of the Android development team
- **Professor Owen Astrachan**, Professor of the Practice of Computer Science at Duke University
- **Larry Page**, Google co-founder, Executive Chairman
- **Dr. Greg Leonard**, economics expert

Trial Witnesses for Oracle

- **Safra Catz**, Oracle co-CEO
- **Edward Screven**, Oracle chief corporate architect

- **Mark Reinhold**, Oracle's chief architect for Java SE
- **Professor Douglas Schmidt**, Professor of Computer Science at Vanderbilt University
- **Neil Civjan**, Sun Microsystem's former head of global sales
- **Alan Brenner**, Sun Microsystem's Senior Vice President of client systems from 1997 until 2007
- **Stefano Mazzocchi**, Google engineer who was one of the original Apache Harmony developers
- **Professor Adam Jaffe**, economics expert

APPENDIX C: TIMELINE

1982: Sun Microsystems founded.

1990: Sun Microsystems begins working on a new programming language to replace C++ and C. Team is code-named "Green Project" and isolated from the rest of the company to develop the product. Originally named Oak, product is later renamed Java, after it is repurposed for Web-page programming.

December 1994: Sun Microsystems secretly invites a select group of programmers to test Java.

January 1995: Sun Microsystems launches Java.

March 1995: Microsoft announces "Blackbird," a new web development package.

May 1995: Netscape (computer services company started by Marc Andreessen in 1994, later acquired by AOL) licenses Java for the Navigator browser.

March 1996: Sun Microsystems enters into Technology License and Distribution Agreement ("TLDA") with Microsoft allowing them to use, modify and adapt Java technology in developing MS Internet Explorer 4.0, and other software products. Microsoft agrees not to fork the code.

1996: Sun Microsystems rolls out first Java Development Kit.

1997: Sun Microsystems approaches International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) to establish Java platform as a formal international standard. Sun ultimately withdraws its application.

October 1997: Microsoft introduces a Microsoft-specific version of Java. Sun sues Microsoft for breach of contract, trademark infringement, copyright infringement, false advertising, and unfair competition. Microsoft threat pushes Sun to pursue an open Java development strategy which encourages widespread adoption as well as adherence to the WORA principle.

1998: Sun Microsystems releases the Java 2 Standard Edition Platform, and gradually expands the number of API packages, classes, and methods over the following years.

1998: Sun Microsystems establishes Java Community Process to allow users to participate in development of Java.

September 1998: Google is founded.

1999: Sun Microsystems develops Java 2 Micro Edition (J2ME) for cell phones.

2000: Dot com bubble bursts.

October 2000: Google launches its AdWords program.

2001: Sun Microsystems and Microsoft settle their litigation. Microsoft agrees to pay Sun $20 million and is permanently barred using "Java compatible" trademarks on its products.

2001: Google names Eric Schmidt, Sun Microsystem's former CTO, as its CEO.

October 2003: Andy Rubin, former Apple engineer and designer of the T-Mobile Sidekick, co-founds Android.

2004: Silicon Valley recovers from Dot Com bubble burst, but Sun Microsystems' hardware business continues to languish. Sun cancels major processor projects, closes one of its two major factories, and initiate a series of layoffs.

2004: Google goes public.

2005: Google acquires Android for $50 million and puts Rubin in charge of mobile platform development project.

2005: Google seeks a permissive (non-GPL) open source Java 2 Platform, Micro Edition JVM license with Sun Microsystems.

2006: To expand Java's reach, Sun licenses Java, including its Standard Edition, Enterprise Edition, and Micro Edition, under the GNU GPLv2.

2006: Jonathan Schwartz takes over Sun Microsystems CEO position from co-founder McNealy.

2006: Google-Sun Java licensing negotiations break down and Google decides to develop a clean room version of Java APIs.

2007: Java is used in 5.5 billion devices, and there are 6 million Java developers. In order to reflect this growing focus on Java, Sun Microsystems changes its Nasdaq Stock Market ticker from SUNW to JAVA.

January 2007: Steve Jobs releases Apple's iPhone.

November 2007: Rollout of Android platform begins.

2008: Sun Microsystems' market value falls 80% between November 2007 and November 2008.

2008–2009: Android products move onto the market, but the iPhone remains dominant.

April 2009: Oracle successfully bids $7.4 billion to purchase Sun Microsystems.

August 2010: Oracle sues Google in Northern District of California alleging that Android infringed seven utility patents and copyrights in the code, documentation, specifications, libraries, and other materials that comprise the Java platform. Oracle seeks a permanent injunction and damages.

September 2011: Judge Alsup rejects Google's motion for summary judgement on copyright cause of action.

April 2012: First district court trial begins with copyright phase. Jury deliberations end with finding that Android infringes Java copyright, but jury hangs on fair use.

May 2012: First district court trial continues with patent phase. Jury rules that Google did not infringe the seven asserted claims of the two patents at issue. Judge Alsup rules on post-trial motions that Java API declarations are uncopyrightable and dismisses case.

October 2012: Oracle and Google appeal district court decisions to the U.S. Court of Appeals for the Federal Circuit.

May 2014: U.S. Court of Appeals for the Federal Circuit reverses district court's determination that the structure, sequence, and organi-

zation of the 37 Java APIs were not copyrightable and remands the fair use issue for retrial with revised jury instructions.

October 2014: Google seeks to challenge the Federal Circuit's reversal by filing a petition for a writ of certiorari with the U.S. Supreme Court.

June 2015: After consulting the Solicitor General, who recommends against granting review on prudential grounds, and also sides with Oracle on substantive grounds, the Supreme Court denies review.

May 2016: Judge Alsup conducts fair use re-trial. Jury rules for Google.

October 2016: Oracle appeals second trial verdict.

APPENDIX D: THE 37 JAVA API PACKAGES IMPLEMENTED IN ANDROID

37 Java API Packages Implemented in Android	
Java API Packages	**Description**
java.awt.font	Provides classes and interface relating to fonts.
java.beans	Contains classes related to developing *beans* — components based on the JavaBeans™ architecture.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.net	Provides the classes for implementing networking applications.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO (Non-blocking I/O) packages. Non-blocking I/O is a collection of Java programming language APIs that offer features for intensive I/O operations. https://en.wikipedia.org/wiki/Non-blocking_I/O_(Java).
java.nio.channels	Defines channels, which represent connections to entities that are capable of performing I/O operations, such as files and sockets; defines selectors, for multiplexed, non-blocking I/O operations.
java.nio.channels.spi	Service-provider classes for the java.nio.channels package.
java.nio.charset	Defines charsets, decoders, and encoders, for translating between bytes and Unicode characters.
java.nio.charset.spi	Service-provider classes for the java.nio.charset package.

37 Java API Packages Implemented in Android	
Java API Packages	**Description**
java.security	Provides the classes and interfaces for the security framework.
java. security.acl	The classes and interfaces in this package have been superseded by classes in the java.security package.
java. security.cert	Provides classes and interfaces for parsing and managing certificates, certificate revocation lists (CRLs), and certification paths.
java. security. interfaces	Provides interfaces for generating RSA (Rivest, Shamir and Adleman Asymmetric Cipher algorithm) keys as defined in the RSA Laboratory Technical Note PKCS#1, and DSA (Digital Signature Algorithm) keys as defined in NIST's FIPS-186.
java. security.spec	Provides classes and interfaces for key specifications and algorithm parameter specifications.
java.sql	Provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java™ programming language.
java.text	Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.
java.util	Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).
java.util.jar	Provides classes for reading and writing the JAR (Java ARchive) file format, which is based on the standard ZIP file format with an optional manifest file.
java. util.logging	Provides the classes and interfaces of the JavaTM 2 platform's core logging facilities.
java.util.prefs	This package allows applications to store and retrieve user and system preference and configuration data.
java.util.regex	Classes for matching character sequences against patterns specified by regular expressions.
java.util.zip	Provides classes for reading and writing the standard ZIP and GZIP file formats.

37 Java API Packages Implemented in Android	
Java API Packages	**Description**
javax.crypto	Provides the classes and interfaces for cryptographic operations. The javax prefix is used by the Java programming language for a package of standard Java extensions. These include extensions such as javax.servlet, which deals with running servlets, and javax.jcr, which deals with the Java Content Library.
javax.crypto.interfaces	Provides interfaces for Diffie-Hellman keys as defined in RSA Laboratories' PKCS #3.
javax.crypto.spec	Provides classes and interfaces for key specifications and algorithm parameter specifications.
javax.net	Provides classes for networking applications.
javax.net.ssl	Provides classes for the secure socket package.
javax.security.auth	This package provides a framework for authentication and authorization.
javax.security.auth.callback	This package provides the classes necessary for services to interact with applications in order to retrieve information (authentication data including usernames or passwords, for example) or to display information (error and warning messages, for example).
javax.security.auth.login	This package provides a pluggable authentication framework.
javax.security.auth.x500	This package contains the classes that should be used to store X500 Principal and X500 Private Credentials in a *Subject*.
javax.security.cert	Provides classes for public key certificates.
javax.sql	Provides the API for server side data source access and processing from the Java™ programming language.

Source: Java™ Platform, Standard Edition 7 API Specification: https://docs.oracle.com/javase/7/docs/api/

APPENDIX E: 2016 FAIR USE TRIAL SUMMARY
(listed in presentation order)

May 9, 2016: Second district court trial begins under Judge Alsup.

Google's Case

- **Eric Schmidt** discusses Sun's encouragement of Java adoption as well as his understanding that Google was free to use the Java APIs without a license.
- **Jonathan Schwartz** explains that Java had always been free and open. Java APIs are free for others to use and independently implement.
- **Andy Rubin** explains his understanding that Android could not use the Java trademarks without a license, but admits that his team could independently implement the Java APIs.
- **Larry Ellison** (via video deposition) denies saying he found Android's use of Java flattering.
- **Joshua Bloch**, a former Sun employee who became Google's "Java guru," explains that the goals of API design are to make them concise and difficult to misuse, and that Sun desired to make the Java APIs widely available.
- **Donald Smith** (via video deposition), a designated Oracle representative, waffles on whether the Java language and APIs are inseparable.
- **Simon Phipps**, Sun's former Chief Open Source Officer and former President of the Open Source Initiative, testifies that Sun had not taken actions to stop other projects that used Java APIs.
- **Daniel Bornstein**, a key member of the Android development team, explains that Java declarations "A-OK to use."
- **Professor Owen Astrachan**, Professor of the Practice of Computer Science at Duke University, instructs the court on API design, the distinction between declaring and implementing code, and the importance of consistent functional labels in programming.

Oracle's Case

- **Safra Catz**, Oracle co-CEO, explains the importance of intellectual property protection to support Oracle's $5.5 billion annual investment in research and development, and discusses how Android's forking of Java code had undermined Oracle's licensing strategy.

- **Edward Screven**, Oracle's Chief Corporate Architect, testifies that Android was the only unlicensed use of Apache Harmony.
- **Mark Reinhold**, Oracle's Chief Architect for Java SE, explains that APIs for Java ME (for feature phones) contain the same structure, sequence, and organization as those for Java SE (for desktop computers).
- **Douglas Schmidt**, Professor of Computer Science at Vanderbilt University, puts into context Google's claim that the Java declaratory code represented less than one-tenth of one percent of Android's fifteen million lines of code by illustrating that more than sixty percent of the Android code was copied from third parties.
- **Neil Civjan**, Sun's head of global sales, characterizes the effect of Android's release on Sun's licensing business as "devastating."
- **Alan Brenner**, Sun's Senior Vice President of client systems from 1997 until 2007, corroborates Civjan's testimony.
- **Stefano Mazzocchi**, Google engineer who was one of the original Apache Harmony developers, rebuts Google's argument that Sun acceded to others' use of the Java APIs.
- **Professor Adam Jaffe**, an economics expert, explains network effects and his conclusion that Android was not transformative from an economic perspective.

Google's Rebuttal

- **Larry Page**, Google's co-founder and CEO, testifies that Google never believed that it needed a license for the Java APIs because they were "free and open."
- **Dr. Greg Leonard**, an economics expert, responds to Professor Jaffe's testimony. Dr. Leonard concludes that Android did not have any impact on licensing Java ME because feature phones were not substitutes for smartphones.
- **Professor Owen Astrachan**, is not surprised at Android's failure to operate with the Java declaring code removed and summarizes Google's approach to designing Android.

May 9, 2016: Jury finds that Google's use of Java was fair use.

June 8, 2016: Judge Alsup denies Oracle's two Rule 50 motions for judgment as a matter of law.

September 20, 2016: Judge Alsup denies Oracle's motion for a new trial.

October 26, 2016: Oracle files appeal with U.S. Court of Appeals for the Federal Circuit. The appeal is pending.