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

PLATFORMS AND INTEROPERABILITY IN *ORACLE V. GOOGLE*

*Joseph Gratz & Mark A. Lemley**

TABLE OF CONTENTS

I. INTRODUCTION 603
II. COPYRIGHT LAW ENCOURAGES INTEROPERABILITY 604
III. INTEROPERABILITY PROMOTES INNOVATION 609
IV. CONCLUSION 613

I. INTRODUCTION

Copyright exists to promote creativity. As a result, the copyright laws strike a series of delicate balances in order to protect different groups of creators. Too much protection for one group thwarts the creative innovation of others. Achieving the right balance is particularly important when it comes to software because software's interactive nature makes the risks of overprotecting existing software particularly great.

Oracle v. Google presents a useful lens to see how this balance is struck. As Peter Menell has documented in detail,[1] the case concerns the extent and limits of interoperability between platforms. In particular, the ultimate outcome of the case will help decide whether the future of phone platforms is open or closed. That in turn has significant implications for innovations written for those platforms.

The Ninth Circuit, like others, has emphasized the importance of interoperability in computer software copyright cases. It has repeatedly held that parties are free to copy the elements of a computer interface necessary to write new and different programs that work with the plaintiff's existing program.[2] The Federal Circuit will nominally apply Ninth Circuit law in *Oracle v. Google*. How it does so will affect the

---

* Joseph Gratz is a partner at Durie Tangri LLP. Mark A. Lemley is a Professor at Stanford Law School and a partner at Durie Tangri LLP.

We wish to thank Rose Hagan and Pam Samuelson for comments on an earlier draft. This article is based on an amicus brief we filed in *Oracle v. Google* on behalf of Engine Advocacy, the App Developers Alliance, and Github, but we have not been paid for this work and our opinions here are entirely our own.

1. Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. (SPECIAL ISSUE) 305 (2018). For other detailed discussion of the case, see Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Test for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215 (2016).

2. *See infra* notes 13–28 and accompanying text.

future of software innovation not just on the Android platform but in "walled gardens" throughout the Internet.[3]

Software companies, and startups in particular, rely on interoperability to build new and innovative products. Without it, developers would be at the mercy of proprietary platforms written in specific, rapidly obsolete computer languages and without the ability to create new and innovative products that are broadly accessible to consumers.[4] The result of such a balkanized regime would be significantly less creativity — the very opposite of what copyright law is designed to achieve. The freedom to interoperate is particularly important in software copyright because copyright in software is more likely than other copyrights to confer control over a market.

II. COPYRIGHT LAW ENCOURAGES INTEROPERABILITY

Debates over interoperability have a long history in software copyright law. The basic contours of that law were established a quarter century ago. At that time, both Oracle and its predecessor Sun lauded the benefits of interoperability. In a brief filed by the American Committee for Interoperable Systems, a trade association that claimed both Sun and Oracle as members in the 1990s, both companies argued that copyrights over application program interfaces (APIs) should not be used to prevent the creation of interoperable programs. They wrote:

> If the developer of one part of the environment can use copyright law to prevent other developers from writing programs that conform to the system of rules governing interaction with the environment — interface specifications, in computer parlance — the first developer could gain a patent-like monopoly over the system without ever subjecting it to the rigorous scrutiny of a patent examination.[5]

Things have changed, as one of the authors worried they might.[6] Oracle's effort to prevent interoperability in *Oracle v. Google* is par-

3. *See, e.g.*, Dan Hunter, *Walled Gardens*, 62 WASH. & LEE L. REV. 607 (2004); Salil K. Mehra, *Paradise is a Walled Garden? Trust, Antitrust, and User Dynamism*, 18 GEO. MASON L. REV. 889 (2012); Greg Lastowka, *Walled Gardens and the Stationer's Company 2.0*, (2013), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2204465 [https://perma.cc/GC6A-423K].

4. *See, e.g.*, Mark A. Lemley & David McGowan, *Legal Implications of Network Economic Effects*, 86 CAL. L. REV. 479 (1998) (discussing problems with lock-in in computer design).

5. Brief for ACIS and CCIA as Amici Curiae Supporting Respondent at *4–5, Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996), (1995 WL 728487).

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

ticularly ironic because Java was itself developed as a way of creating interoperability across platforms. But because Java was not released as open source software, Lemley & McGowan worried in 1998 about the possibility that Sun would try to close the Java standard to others to reap the benefits of widespread adoption.[7] And indeed that is what happened after Oracle bought Sun.

Whether Oracle *can* close the standard is another matter — a legal one. To some extent those issues are a function of the history of Java, which might create contract or estoppel rights to continued access to Java APIs on behalf of existing users or perhaps even the public. A failure to honor those rights might even run afoul of the anti-antitrust laws, though courts have properly made proof of such an antitrust violation difficult. Those issues are discussed elsewhere, and we shall not focus on them here.[8]

Rather, our focus is on the role of interoperability in copyright law. If Oracle has no power under copyright to restrict the writing of interoperable programs, either applications programs or platforms, its desire to close off Java to competition will not matter much. Companies that want to write interoperable programs will be able to reverse engineer the Java code or copy publicly exposed APIs in order to do so.

Software copyright law has long favored interoperability. In many cases it has done so by denying protection altogether to elements of computer programs that exist only for purposes of interoperability, like APIs.[9] The Federal Circuit's prior decision in *Oracle v. Google*

---

7. *Id.*

8. *Id.*; Mehra, *supra* note 3.

9. *See, e.g.*, DSC Commc'ns v. DGI Tech., 81 F.3d 597, 601 (5th Cir. 1996); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1539 n.18 (11th Cir. 1996); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 821 (1st Cir. 1995) (Boudin, J., concurring), *aff'd*, 116 S. Ct. 804 (1996); Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1527–28 (9th Cir. 1992); Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 843–44 (Fed. Cir. 1992); Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 270 (5th Cir. 1988); Mitel Inc. v. Iqtel Inc., 896 F. Supp. 1050 (D. Colo. 1995); *see also* JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL: INTELLECTUAL PROPERTY AND INTEROPERABILITY IN THE GLOBAL SOFTWARE INDUSTRY (1995); Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091 (1995); Lawrence D. Graham & Richard O. Zerbe Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure*, 22 RUTGERS COMPUT. & TECH. L.J. 61 (1996); Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 U. DAYTON L. REV. 975, 1016–18 (1994); David A. Rice, *Sega and Beyond: A Beacon for Fair Use Analysis . . . At Least as Far as It Goes*, 19 U. DAYTON L. REV. 1131, 1168 (1994); *cf.* Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1 (1995) (noting that courts are reaching results that permit interoperability, albeit through various means). For a discussion of the various circumstances in which courts have upheld interoperability defenses, and a suggested refinement of the *Altai* test, see Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1297 (2017).

foreclosed that approach here.[10] It has (justly) been criticized for that.[11]

But even if the Java APIs are copyrightable, that does not mean that their use is copyright infringement. As described below, the Ninth Circuit has repeatedly interpreted the Copyright Act's fair use doctrine to protect the right of third parties to copy APIs when necessary to make their products work with products made by the copyright owner or others. That is true even when the use of the API requires copying the computer code itself, not just the higher-level functional aspects of the API. And it is true even if the defendant copies the API in order to compete directly with the plaintiff by producing a compatible system.[12]

In *Sega Enterprises Ltd. v. Accolade, Inc*.,[13] for example, Accolade wanted to make video games compatible with Sega's game console over Sega's objection. To make its games run on Sega's platform, Accolade copied the entirety of Sega's computer code in order to "reverse engineer" the code and extract only the APIs — the portions necessary to ensure compatibility. The Ninth Circuit held that was a fair use even though it involved copying of the entirety of the code, because making that copy was necessary to get access to the interface components — which the Ninth Circuit found to be "unprotectable."[14] The Court emphasized that "because Accolade has a legitimate interest in gaining such access (in order to determine how to make its cartridges compatible with the Genesis console)," its copying of the code to replicate the interface components was a fair use.[15]

The fact that Accolade sought to write its own original programs, not to copy Sega's programs, loomed large in the Ninth Circuit's analysis:

> Accolade copied Sega's software solely in order to discover the functional requirements for compatibility with the Genesis console — aspects of Sega's programs that are not protected by copyright. With respect to the video game programs contained in Accolade's game cartridges, there is no evidence in the record that Accolade sought to avoid performing its own creative work. Indeed, most of the games that

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

11. *See* Menell, *supra* note 1; Samuelson, *supra* note 9.

12. In a companion paper, Pamela Samuelson and Clark Asay point out other flaws in Oracle's fair use analysis, including the utilitarian nature of APIs and a misunderstanding of the role of "good faith" in fair use. *See* Pamela Samuelson & Clark D. Asay, *Saving Software's Fair Use Future*, 31 HARV. J.L. & TECH. (SPECIAL ISSUE) 535 (2018).

13. 977 F.2d 1510 (9th Cir. 1992), *as amended* (Jan. 6. 1993).

14. *Id.*

15. *Id.* at 1520.

> Accolade released for use with the Genesis console were originally developed for other hardware systems . . . [A]lthough Accolade's ultimate purpose was the release of Genesis-compatible games for sale, its direct purpose in copying Sega's code, and thus its direct use of the copyrighted material, was simply to study the functional requirements for Genesis compatibility so that it could modify existing games and make them usable with the Genesis console . . . On these facts, we conclude that Accolade copied Sega's code for a legitimate, essentially non-exploitative purpose . . .[16]

Nor was the court troubled that Accolade engaged in verbatim copying of some program interfaces in order to achieve that legitimate compatibility purpose:

> [C]omputer programs are, in essence, utilitarian articles — articles that accomplish tasks. As such, they contain many logical, structural, and visual display elements that are dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry demands . . . 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 infringement.[17]

In *Sony Computer Entertainment, Inc. v. Connectix Corp.*,[18] the Ninth Circuit went further, holding that it was fair use to create an emulator of the Sony game console — copying the code not just to reverse engineer it but to test how programs worked with it — because the purpose was to produce a new product that worked with the old system:

> We find that Connectix's Virtual Game Station is modestly transformative. The product creates a new platform, the personal computer, on which consumers can play games designed for the Sony PlayStation. This innovation affords opportunities for game play in new environments, specifically an-

16. *Id.* at 1522–23 (citations omitted).
17. *Id.* at 1524 (internal quotation marks omitted) (citations omitted).
18. 203 F.3d 596 (9th Cir. 2000).

> ywhere a Sony PlayStation console and television are not available, but a computer with a CD-ROM drive is. More important, the Virtual Game Station itself is a wholly new product, notwithstanding the similarity of uses and functions between the Sony PlayStation and the Virtual Game Station.[19]

For the same reason, the fact that Sony might lose sales to the Connectix system did not militate against fair use on the fourth factor. "[B]ecause the Virtual Game Station is transformative, and does not merely supplant the PlayStation console, the Virtual Game Station is a legitimate competitor in the market for platforms on which Sony and Sony-licensed games can be played[,]" so the loss of market share was not attributable to copyright infringement, but to legitimate competition.[20]

Judged against this Ninth Circuit precedent, Google has a strong claim to fair use. Google did not simply copy Java. Instead, it took only what was necessary to make Java API calls usable by programs running on the Android phone system. That was an innovation. Java was designed for desktop computers and was unsuited for use on phones.[21] Neither Sun nor Oracle succeeded in creating a smartphone operating system using Java (or Java API calls).[22] Google created the first mainstream open-source mobile platform, Android. The Android platform was "revolutionary" and "completely different from any other approach."[23] That new platform is overwhelmingly Google's work, not Sun's or Oracle's. Android includes 15 million lines of code, only a tiny fraction of which are shared in common with Java.[24] Nor did Android even copy all of Java's APIs. Rather, it used declarations from only 37 of the 166 Java API packages, while Google wrote its

---

19. *Id.* at 606.

20. *Id.* at 607.

21. Record App'x. 51938:6–19, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17–1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

22. Record App'x. 50559–50560 at 560:17–561:4, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016); Record App'x. 51896:12–17, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

23. Record App'x. 50346–50347 at 347:14–348:7, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016); Record App'x. 50347–50348 at 348:21–349:1, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

24. Record App'x. 51247:4–7, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

own code to implement those functions.[25] Those 37 APIs are, by Oracle's own admission, "not separable" from the Java programming language and are "fundamental" to implementing Java.[26] So Oracle's argument that implementing those APIs is not fair use as a matter of law is tantamount to arguing that interoperability is not fair use as a matter of law.[27]

Sega and *Sony* hold that a company is free to copy the entirety of a computer program in order to build a compatible product, even where that compatible product copies API code directly in the final product and even where that final product competes directly with the plaintiff. Google's use of selected Java APIs, which did not involve copying of the code in the final product and did not involve a competing product at all, seems by comparison easy to justify as fair use under the Ninth Circuit's case law promoting interoperability.[28]

III. INTEROPERABILITY PROMOTES INNOVATION

The legality of copying APIs and other interface components is well-established, and has been for a quarter-century. This is true not only in the Ninth Circuit but in all other circuits that have considered the issue.[29] In addition, Congress endorsed interoperability when it enacted the Digital Millennium Copyright Act. While that Act made it illegal to circumvent a technical protection measure that controlled access to a copyrighted work, Congress was careful not to prohibit circumvention "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 pro-

25. Record App'x. 51098–51099 at 1097:19–1098:11, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

26. Record App'x. 51014, Oracle Am., Inc. v. Google Inc., (Fed. Cir. 2017) (No. 17-1118) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016).

27. Oracle argued at trial that this was not a matter of "interoperability," since in the end Java programs could not be run unmodified on Android. We are unpersuaded by this distinction. The justification for interoperability is as strong where the use merely *facilitates* the porting of programs from one platform to another, even if humans are involved in completing that process.

28. *See* Clark D. Asay, *Transformative Use in Software*, 70 STAN. L. REV. ONLINE 9 (2017) (arguing that accepting Oracle's argument on appeal against the jury's finding of fair use would mean that fair use rarely if ever applies in the software context).

29. *See, e.g.*, Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 710–15 (2d Cir. 1992); Lexmark Int'l, Inc. v. Static Control Components, Inc., 387 F.3d 522, 542 (6th Cir. 2004), *reh'g en banc denied* (2005); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–19 (1st Cir. 1995), *aff'd*, 516 U.S. 233 (1996); *see also id.* at 821 (Boudin, J., concurring); Assessment Techs. of WI, LLC v. WIREdata, Inc., 350 F.3d 640, 644–45 (7th Cir. 2003); Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1374–76 (10th Cir. 1997); Pamela Samuelson & Suzanne Scotchmer, *The Law and Economics of Reverse Engineering*, 111 YALE L.J. 1575, 1621–26 (2002).

grams . . .”[30] Thus, a reversal of the trial court’s fair use finding in *Oracle v. Google* would conflict not just with the great weight of judicial decisions, but also with policy choices made by Congress.

We think such a change would be unwise. Computer programmers and software companies rely on that settled law.[31] Computers and the Internet work because different programs and devices can communicate with each other. Interoperability makes that possible.[32] Interoperability is the reason you can read a web site regardless of what Internet browser you use.[33] Interoperability is the reason you can read documents on a PC even though someone wrote them on a Mac.[34] Interoperability is the reason messages can pass from phone to computer to tablet and back again.[35]

Indeed, the law’s solicitousness to copying for the purpose of interoperability is the reason we have a vibrant and competitive personal computer industry in the first place. As has been recounted in detail by others,[36] IBM controlled the market for PC-compatible computers in the early 1980s through its control of the IBM PC BIOS — the set of software that provides an API for software, including the operating system, to communicate with the computer’s processor. Software written for the IBM PC was written to communicate using APIs provided by the IBM PC BIOS. In order to run that software, competing

---

30. 17 U.S.C. § 1201(f)(1) (1998).

31. *See, e.g.*, Brief of Computer Scientists as Amici Curiae Supporting Defendant-Appellee at 1–3, Oracle Am., Inc. v. Google Inc., No. 17-1118 (Fed. Cir. 2017) (pending appeal from 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) (No. C 10-03561 WHA), docketed Oct. 28, 2016), https://www.eff.org/files/2017/05/31/2017.05.30_computer-scientists-fair-use-amicus-brief_oracle_v_google.pdf [https://perma.cc/HFC3-UJF2] (brief of seventy-six widely recognized computer scientists arguing that the “software industry has long relied on and benefitted from the open nature of application programming interfaces”).

32. *Id.* at 10–14.

33. *See Cross-Browser Compatibility Tutorial: Use JS for Cross-Browser Compatibility*, YOUTUBE (Sept. 30, 2015), https://www.youtube.com/watch?v=FGAV4UMvedk (last visited Jan. 26, 2018); Richard Cornford, *Browser Detection (and What to Do Instead)*, JIBBERING, http://jibbering.com/faq/notes/detect-browser/ [https://perma.cc/N53D-VCEH]; *What is JavaScript?*, MOZILLA, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript [https://perma.cc/A2YS-CH7A].

34. Gina Trapani, *The Complete Guide to Mac/Windows Interoperability*, LIFEHACKER (Oct. 19, 2007), https://lifehacker.com/311618/the-complete-guide-to-macwindows-interoperability [https://perma.cc/43MM-G7DP]; *see also* Erik Eckel, *Mac vs. Windows Incompatibility Achieves Irrelevance*, TECHREPUBLIC (Feb. 16, 2015), http://www.techrepublic.com/article/mac-vs-windows-incompatibility-achieves-irrelevance/ (last visited Jan. 27, 2018).

35. *See Cross-Platform Interoperability*, JWSECURE.COM, http://www.jwsecure.com/services/cross-platform-interoperability/ [https://perma.cc/U5GM-QMXU]; *What is Cross-Platform Software?*, BOBOLOGY, https://www.bobology.com/public/What-is-CrossPlatform-Software.cfm [https://perma.cc/DSM6-3BAC]; *see also* Daniel Nations, *How to Develop for iOS, Windows and Mac at the Same Time*, https://www.lifewire.com/develop-for-ios-android-windows-mac-1994294 [https://perma.cc/KQ7W-ESG9].

36. *See, e.g.*, Russell Moy, *A Case Against Software Patents*, 17 SANTA CLARA COMPUT. & HIGH TECH. L.J. 67, 71 (2000); CHARLES FERGUSON & CHARLES MORRIS, COMPUTER WARS 52–53 (1993).

PC makers needed to provide their own BIOS that could use those APIs. In 1984, a company called Phoenix Technologies reimplemented the IBM PC BIOS API in its own original software code through a "clean room" reimplementation, copying only the elements necessary for compatibility.[37] As with Java, those elements included a hierarchy of commands — for example, all calls beginning with "0x10" related to video services, and within that category the call "0x10 0x9H" would write a particular letter to the screen.[38] IBM did not take legal action against Phoenix, and the availability of Phoenix BIOS led to a proliferation of IBM PC-compatible "clone" computers from Compaq, Dell, and others.[39]

Some of this is due to software copyright holders voluntarily opening standards, either from the beginning or due to pressure from third parties. Many times, companies consciously allow others to make compatible programs.[40] For example, the JavaScript standard — which, despite the name, bears no relationship to Java — was published as an open standard.[41] In some instances, standards that remain closed are reverse-engineered and published by others, so that compatible programs or products can be marketed.[42] Such third-party publication sometimes leads the copyright holder to open an otherwise closed standard.[43] In this way, the fact that copyright law permits reverse engineering for purposes of interoperability takes away a copyright owner veto and thereby allows a wider range of experimentation.

The law's openness to use of APIs has led, over time, to some unusual turnabouts. For example, in 1993, open-source developer Andre Julliard released WINE, a program that allows Windows applications to run on computers that use the Linux operating system. It does

---

37. James Langdell, *Phoenix Says Its BIOS May Foil IBM's Lawsuits*, PC MAGAZINE 56 (July 10, 1984), https://books.google.com/books?id=Bwng8NJ5fesC&lpg=PA6&pg=PA56#v=onepage&q&f=false (last visited Jan. 26, 2018).

38. INT'L BUS. MACHS. CORP., IBM PERSONAL SYSTEM/2 AND PERSONAL COMPUTER BIOS INTERFACE TECHNICAL REFERENCE 2–17 (1987), http://www.nj7p.org/Computers/IBM%20PC/work/BIOS_Interface_Technical_Reference.pdf [https://perma.cc/V7NZ-L3FN].

39. *Send in the Clones*, COMPUT. HISTORY MUSEUM, http://www.computerhistory.org/revolution/personal-computers/17/302 [https://perma.cc/468T-RGUB].

40. HTML, CSS, and ECMAScript are examples.

41. *ECMAScript® 2017 Language Specification (ECMA-262, 8th ed., June 2017)*, ECMA INT'L, https://www.ecma-international.org/ecma-262/8.0/index.html [https://perma.cc/P3JR-PB29].

42. *Apple Accessory Protocol*, NUXX, https://nuxx.net/wiki/Apple_Accessory_Protocol [https://perma.cc/UNA7-75UT] (disclosing reverse-engineered protocol for communicating with an iPod).

43. *See, e.g.*, *Office File Formats*, MICROSOFT, https://msdn.microsoft.com/en-us/library/office/cc313118(v=office.12).aspx [https://perma.cc/UGC7-PNLY] (disclosing Microsoft Office file formats, which were previously proprietary); *Using the HomeKit Accessory Protocol Specification (Non-Commercial Version)*, APPLE, https://developer.apple.com/support/homekit-accessory-protocol/ [https://perma.cc/WP4S-ERF7] (disclosing Apple HomeKit Accessory Protocol, which was previously proprietary).

so by translating, in real time, calls to Windows APIs into the corresponding calls in the POSIX APIs that Linux uses. To do so, it must use the same hierarchy of function names, just as Android does with respect to Java. Unlike Oracle, however, Microsoft has not sought to halt the distribution or use of WINE, perhaps recognizing that reimplementing APIs with the same names does not violate any of its rights.

And interoperability is a two-way street. Just as Linux users found it useful to be able to run some Windows programs they were familiar with, Windows users — particularly developers — found it useful to be able to run some Linux programs they were familiar with. In 2016, Microsoft released the Windows Subsystem for Linux, which provides, in essence, the inverse of WINE: it allows Linux programs to run on Windows, translating API calls in real time to allow the programs to run unmodified. Microsoft engaged in a "clean room" reimplementation of the Linux kernel APIs to ensure that only the API structure, and not any of the implementing code, was copied.[44]

Interoperability is also critical to the development of the new Internet of Things ("IoT") that connects a wide array of devices beyond computers.[45] IoT by definition depends on autonomous communication amongst a wide range of devices.[46] That cannot happen without interoperable standards in the IoT market.[47] Fragmentation of the market due to competing, proprietary standards will severely curtail the value of IoT as a whole.[48] Considering VCs invested more than $1

---

44. *Windows Subsystem for Linux Overview*, MICROSOFT (Apr. 22, 2016), https://blogs.msdn.microsoft.com/wsl/2016/04/22/windows-subsystem-for-linux-overview/ [https://perma.cc/4645-ZFE8] ("The drivers do not contain code from the Linux kernel but are instead a clean room implementation of Linux-compatible kernel interfaces.").

45. *See, e.g.*, Lu Tan, *Future Internet: The Internet of Things*, 5 INST. ELEC. & ELECS. ENG'R 376, 379 (2010) ("Only if we can solve the interoperability problem we can have a real Internet of Things."); *Developing the Interoperable Internet of Things*, OPEN CONNECTIVITY FOUND. (June 27, 2017), https://openconnectivity.org/blog/developing-interoperable-internet-things [https://perma.cc/UFU5-H5XG]; Gary Eastwood, *IoT's Interoperability Challenge*, NETWORKWORLD (July 5, 2017, 6:03 AM), https://www.networkworld.com/article/3205207/internet-of-things/iots-interoperability-challenge.html [https://perma.cc/U6YA-4HQQ]; Giancarlo Fortino et al., *Interoperability in the Internet of Things*, COMPUTER.ORG (Dec. 2016), https://www.computer.org/web/computingnow/archive/interoperability-in-the-internet-of-things-december-2016-introduction [https://perma.cc/S9UH-GJNA]; *Interoperability: The Challenge Facing the Internet of Things*, PROPHET, https://www.prophet.com/thinking/2014/02/interoperability-the-challenge-facing-the-internet-of-things/ [https://perma.cc/GVG3-LBNN]; James Manyika et al., *Unlocking the Potential of the Internet of Things*, MCKINSEY & CO. (June 2015), http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world [https://perma.cc/4YW5-3F2N]; Phillip Tracy, *IoT Interoperability: Where it Stands and What Comes Next*, RCRWIRELESS NEWS (Oct. 31, 2016), https://www.rcrwireless.com/20161031/internet-of-things/iot-interoperability-tag31-tag99 [https://perma.cc/44JV-GBF8].

46. *See* Tan, *supra* note 45.

47. *Id.*

48. *Id.*

billion in IoT startups in 2016,[49] interoperability in the IoT sector has substantial economic consequences. The importance of interoperability in the software and Internet environments has been so clearly demonstrated, and so widely accepted, that most IoT programmers are writing only to open standards in the first place.[50] But even where parties contract for interoperability, for instance by using open source software, legal interoperability plays a role. It allows downstream users to avoid an "anticommons" of overlapping and potentially conflicting contractual commitments.[51]

Interoperability is particularly important to startups. Companies that develop apps for mobile phones are often small. They may not have the ability to write several different versions of a program from scratch, one for each hardware platform or incompatible programming language — much less to separately negotiate agreements with each such platform provider in the economy. By allowing an app developer to reach the widest possible market, legal protection for interoperability increases the number and availability of creative new works produced each year. It also ensures that no one company, no matter how dominant its platform, gets to decide what web pages you can access, what files you can share, or what programs you can download.

Without the security to investigate and use APIs, software developers would be at the mercy of platform and programming giants who could decide whether, when, and how anyone could write or use a computer program that ran on their system. Startups will not invest in new products — for mobile phones or video games or the Internet of Things — without confidence that their products will work on the dominant platforms. That is why the risk of overprotecting copyright is so much greater in software than in other areas. Giving too much protection to a song may incrementally discourage the creation of somewhat similar songs. Giving copyright owners control over interoperability risks shutting down the software development ecosystem altogether.

IV. CONCLUSION

The Java slogan was "Write Once, Run Anywhere." Copyright law allows software developers to do just that: write a single program

---

49. Mikey Tom, *IoT Breakdown: VCs Betting Billions on the Connected World*, PITCHBOOK (Dec. 7, 2016), https://pitchbook.com/news/articles/iot-breakdown-vcs-betting-billions-on-the-connected-world [https://perma.cc/772B-HEPU].

50. Brian Ray, *Open Source Software and Hardware for the Internet of Things*, IOT FOR ALL (Jul. 8, 2017), https://medium.com/iotforall/open-source-software-and-hardware-for-the-internet-of-things-eca2aa728fa4 [https://perma.cc/9ZTD-TABR].

51. Clark D. Asay, *Software's Copyright Anticommons*, 66 EMORY L.J. 265 (2017); *see also* Clark D. Asay, *Copyright's Technological Interdependencies*, 18 STAN. TECH. L. REV. 189 (2015).

that works on multiple platforms. That in turn encourages more creative works. As the Ninth Circuit explained in *Sega*:

> Accolade's identification of the functional requirements for Genesis compatibility has led to an increase in the number of independently designed video game programs offered for use with the Genesis console. It is precisely this growth in creative expression, based on the dissemination of other creacreative works and the unprotected ideas contained in those works, that the Copyright Act was intended to promote.[52]

Three decades ago, Peter Menell explained the risks of overprotecting software once it becomes a standard.[53] As he notes in his article in this issue, *Oracle v. Google* raises that old concern anew.[54] To use copyright, a law designed to promote creativity, to stifle it instead would be perverse.

52. Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1523 (9th Cir. 1992).

53. Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1050 (1989).

54. Menell, *supra* note 1.