Harvard Journal of Law & Technology
Volume 32, Number 1 Fall 2018

APPLICATION PROGRAMMING INTERFACES AND THE STANDARDIZATION-VALUE APPROPRIATION PROBLEM

*Parth Sagdeo**

TABLE OF CONTENTS

---

* Law Clerk, U.S. District Court for the District of Delaware; J.D., Harvard Law School, 2018; M.S. in Electrical & Computer Engineering, University of Illinois at Urbana-Champaign, 2013; B.S. in Electrical Engineering & Computer Science, University of California, Berkeley, 2010.

I would like to thank Prof. Ted Sichelman for teaching the fascinating class that led to this Note, Prof. William Fisher for his advice, and the staff of the Harvard Journal of Law and Technology, especially Article Editor King Xia, for their thoughtful and thorough editing of this Note.

That being said, all opinions and errors are my own.

I. INTRODUCTION

Application programming interfaces ("APIs") are the electrical sockets of modern software systems. Just as every electric device with a certain type of plug fits every outlet of the corresponding type, APIs allow software written at different times, by different people, and in different organizations to work together seamlessly. They have enabled, for example, millions of applications, worth billions of dollars, to be written for Android, iOS, and Windows by companies other than Google, Apple, and Microsoft, respectively.[1]

An API specifies how software components are supposed to interact with each other.[2] At a high level, this interface includes a list of commands that a first component can use to access functionality in a second component, and includes the specific format in which the first component should give those commands to the second component.[3] Some software components, such as a web browser's user interface, are readily visible to the user.[4] Many more components are hidden but perform key functions. For example, various software components are responsible for sending and receiving web page data over the Internet, parsing that data and rendering it in a graphical format, and managing persistent data (e.g., browser cookies) stored by websites.[5] APIs define the interaction between these components.

---

1. *See, e.g.*, *Number of Apps Available in Leading App Stores*, STATISTA (2018), https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores (last visited Dec. 20, 2018); Sarah Perez, *App Revenue Climbed 35 Percent to $60 billion in 2017*, TECHCRUNCH (Jan. 5, 2018), https://techcrunch.com/2018/01/05/app-revenue-climbed-35-percent-to-60-billion-in-2017 [https://perma.cc/SH7X-RG69].

2. *API (Application Programming Interface) Definition*, TECHTERMS (June 20, 2016), https://techterms.com/definition/api [https://perma.cc/27P6-W7A8].

3. *See id.*

4. *See, e.g.*, *UIKit*, APPLE DEVELOPER DOCUMENTATION, https://developer.apple.com/reference/uikit [https://perma.cc/4ZNC-RNCT].

5. *See, e.g.*, *WebKit*, APPLE DEVELOPER DOCUMENTATION, https://developer.apple.com/reference/webkit [https://perma.cc/H26D-YKNB].

As discussed in Part II of this Note, the creation of APIs can involve considerable creativity and investment of resources. Depending on the circumstances, APIs can be protected by trade secret, patent, and/or copyright law. In practice, however, many APIs cannot be adequately protected by trade secret or patent law. Thus, copyright provides the most reliable means of intellectual property ("IP") protection for APIs.

Part III of this Note notes that IP protection of APIs has drawn criticism for decades. In particular, commentators have identified two features of APIs — network effects and switching costs — that acting together can cause market monopolization and other negative externalities including spurring excessive marketing costs, increasing prices for consumers (generating deadweight losses), and increasing barriers to further innovation.

Part III of this Note contributes to this longstanding discussion by examining the value of API copyrights, and finds that copyright overcompensates API owners in one key dimension: namely, the ability of owners to appropriate user switching costs. This overcompensation is the "standardization-value appropriation problem." This Note argues that preventing API copyright owners from appropriating standardization value averts the above-mentioned harms to general welfare (at least to the extent that they are greater for APIs than for other copyrighted works) and does not disturb the incentive structure underlying copyright.

Part IV of this Note discusses three possible legal regimes that prevent a copyright owner from appropriating standardization value: (1) a regime where APIs are uncopyrightable; (2) a regime where use of an API for interoperability is fair use; and (3) a statutory fixed-rate licensing regime. The Note concludes that all three of these regimes have substantial problems in optimally incentivizing innovation.

Part V of this Note proposes a variable-rate compulsory licensing regime for APIs. Under this regime, API owners must provide a compulsory license to others. The royalty rate for the license is calculated using fair, reasonable, and nondiscriminatory ("FRAND") licensing principles. The Note concludes that such a regime provides the closest-to-optimal incentive for innovation, while having higher but still-manageable transaction costs.

II. FROM API TO IP

This Part provides background on the technical and legal concepts discussed in this Note. Section II.A provides a short technical introduction to APIs. Section II.B discusses the types of intellectual property protection that an API can receive, and the circumstances in which intellectual property protection of APIs is effective.

A. A Short Introduction to APIs

1. Software Libraries as a Means for Abstraction

Virtually all computer programs written today are useless in isolation. Each depends on functionality provided by other software. Much of this functionality is provided by libraries: prewritten code that implements a series of related functions given well-defined inputs.[6] For example, operating system libraries (e.g., those provided by Windows, macOS, and Android) allow programs to communicate over a network, access storage devices, and modify what the computer displays.[7] Most programming languages (e.g., C, Python, and Java) also provide a "standard library" for that language.[8] Standard libraries allow programmers working in that particular language to access a computer's operating system.[9] Standard libraries also provide programmers with premade building blocks that implement commonly used functionality. For instance, Python has libraries for natural language processing, statistics, and bioinformatics.[10]

The use of libraries has many benefits, including increased reliability, more readable code, and faster development of new applications.[11] Most importantly, libraries allow programmers to work at higher levels of abstraction — tying together "building blocks" of functionality rather than having to construct software from scratch. By using libraries built on top of libraries, application developers can create high-level software

---

6. *See Library*, COMPUTER DESKTOP ENCYCLOPEDIA, http://lookup.computerlanguage.com/host_app/search?cid=C999999&term=library [https://perma.cc/HN5G-FG7U] (definition (3)).

7. *See, e.g.*, *Windows API Index*, MICROSOFT (May 30, 2018), https://docs.microsoft.com/en-us/windows/desktop/apiindex/windows-api-list [https://perma.cc/K3JY-B832]; *API Reference*, APPLE DEVELOPER DOCUMENTATION, https://developer.apple.com/reference [https://perma.cc/YGS7-N22E].

8. *See, e.g.*, *C Standard Library Reference Tutorial*, TUTORIALSPOINT, https://www.tutorialspoint.com/c_standard_library [https://perma.cc/AL3J-4ZVB]; *The Python Standard Library*, PYTHON 3.7.1 DOCUMENTATION, https://docs.python.org/3.7/library [https://perma.cc/3C2K-36AL]; *Overview*, JAVA PLATFORM SE 7, https:/docs.oracle.com/javase/7/docs/api [https://perma.cc/5FWZ-QC6S].

9. *See, e.g.*, *Generic Operating System Services*, PYTHON 3.7.1 DOCUMENTATION, https://docs.python.org/3.7/library/allos.html [https://perma.cc/JX28-V9JC]; *C Library Function - System()*, TUTORIALSPOINT, https://www.tutorialspoint.com/c_standard_library/c_function_system.htm [https://perma.cc/AH39-KCJB].

10. *Natural Language Toolkit*, NLTK 3.3 DOCUMENTATION (May 6, 2018), https://www.nltk.org [https://perma.cc/8JWP-ZPQ8]; *9.7. Statistics — Mathematical Statistics Functions*, PYTHON 3.7.1 DOCUMENTATION, https://docs.python.org/3.7/library/statistics.html [https://perma.cc/CAR3-WWTN]; *Biopython*, BIOPYTHON, https://biopython.org/ [https://perma.cc/ZF99-SZ6B].

11. *See* Pierre Mengal, *6 Advantages to Using Third Party Libraries over Developing Your Own*, MINDFULHACKER (Apr. 14, 2012), http://www.mindfulhacker.com/6-advantages-using-third-party-libraries-over-developing-your-own [https://perma.cc/9FHR-K7Z6].

that does not concern itself with specific methods of computation or the manner in which the hardware needs to be used to achieve a specific result. This abstraction is incredibly powerful: it allows, for example, a quickly-written Android application to run on thousands of hardware configurations. Figure 1 shows how a hypothetical Android dialer application called "CustomDialer" can be built on several layers of libraries to leverage library abstraction.

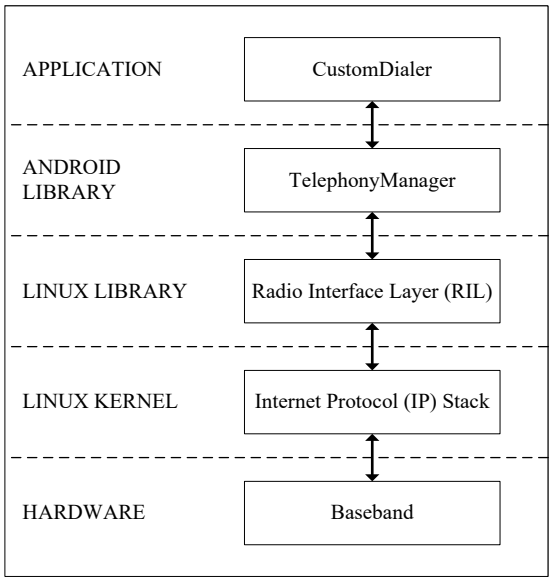

Figure 1: The Libraries that Might Be Used by an Android Dialer Application[12]

2. The Application Programming Interface

An application programming interface ("API") is the interface between an application and a library.[13] It consists essentially of two parts: (1) its declaring code, and (2) its structure, sequence, and organization.[14] "Declaring code" is the code an application developer (i.e., a developer using the API) needs to use to call upon specific functionality in the library.[15] The API's structure, sequence, and organization ("SSO") is es-

12. *See Radio Layer Interface*, ANDROID OPEN SOURCE PROJECT, https://wladimir-tm4pda.github.io/porting/telephony.html [https://perma.cc/3PM4-9HAE].

13. *See* Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222 (3d Cir. 1986).

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

15. *Id.*

sentially the taxonomy under which the declaring code is structured.[16] Figure 2 shows the relationship between an application, an API, and a library. On one side of the interface, an application can use the API without knowing anything about how the underlying library works (i.e., about its implementing code), so long as they conform to the API's declaring code and SSO. On the other side, an API implementer can implement a library in any manner they see fit, so long as they conform to the API's SSO and declaring code. And although an API and a corresponding library are often both created by the same organization, in principle they are completely independent works.[17]

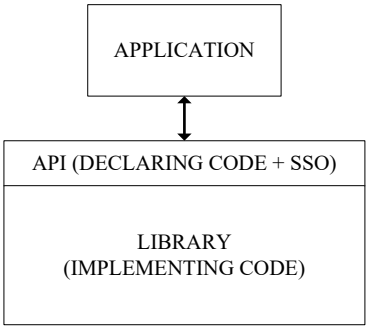

Figure 2: The Relationship Between an Application, an API, and a Library

B. Copyright Law is the Only Existing IP Right That Can Protect Many APIs

The API's relationship with intellectual property law has been evolving for decades.[18] APIs have been held to be patent eligible,[19] patent ineligible,[20] copyright eligible,[21] copyright ineligible,[22] trade secret

16. *Id.*; *see also Whelan*, 797 F.2d at 1248 (introducing the term "structure, sequence, and organization" to describe the organization of an API).

17. The ECMAScript (more commonly referred to as JavaScript) API is an example of an API that is created and implemented by different entities. ECMAScript is defined by Ecma International, a standards body. *See generally Standard ECMA-262*, ECMA INTERNATIONAL (June 2018), https://www.ecma-international.org/publications/standards/Ecma-262.htm [https://perma.cc/H25M-SHE8]. ECMAScript is implemented by every modern web browser, including Chrome, Edge, Firefox, Internet Explorer, and Safari. *JavaScript Versions*, W3 SCHOOLS, https://www.w3schools.com/js/js_versions.asp [https://perma.cc/76YE-YQJ3].

18. *See* Pamela Samuelson, *The Strange Odyssey of Software Interfaces and Intellectual Property Law* 12–13 (Univ. Cal. Berkeley Pub. Law Research Paper No. 1323818, 2009), https://ssrn.com/abstract=1323818 [https://perma.cc/N2TH-YFSA].

19. Apple Inc. v. Motorola, Inc., 757 F.3d 1286, 1307 (Fed. Cir. 2014), *overruled by* Williamson v. Citrix Online, LLC, 792 F.3d 1339 (Fed. Cir. 2015).

20. Allvoice Devs. US, LLC v. Microsoft Corp., 612 F. App'x 1009, 1017 (Fed. Cir. 2015).

eligible,[23] and trade secret ineligible.[24] But, currently, copyright is the only form of intellectual property that can reliably protect publicly available software APIs.

1. Patenting APIs

Most APIs are not patent eligible, especially post-*Alice*.[25] Considering the components of an API — declaring code and its SSO — this is not surprising. Declaring code falls squarely within the printed matter exception to patentability because: (1) source code is printed matter; (2) it is nonfunctional (only the *implementing* code is functional); and (3) it is nonstructural (it has nothing to do with the structure of the computer readable medium it is typically stored on).[26] Because the SSO is how the declaring code is organized, it is essentially a "mere arrangement of printed matter," and so it too is patent ineligible.[27] APIs may become patentable if they are combined with novel hardware elements,[28] though in such cases it is not the API in itself that is protected, but its combination with patent-eligible elements.

2. Protecting APIs as Trade Secrets

APIs can theoretically be protected as trade secrets, but this protection is significantly limited by the secrecy requirement of, and reverse engineering exception to, trade secret law. One of the most important benefits of APIs is the ability for a platform owner (e.g., Apple, Google, or Microsoft) to enable third-parties to develop applications for the platform. To encourage third-party developers, platform owners will not only publish their APIs, but typically provide extensive documentation and tooling on how to use their APIs.[29] In view of this extensive public

21. *Oracle*, 750 F.3d at 1381.

22. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 721 (2d Cir. 1992).

23. Altavion, Inc. v. Konica Minolta Sys. Lab. Inc., 171 Cal. Rptr. 3d 714, 731 (Cal. Ct. App. 2014) (holding that the "detailed design concepts underlying Altavion's [software]" were protectable as a trade secret).

24. *See* DVD Copy Control Ass'n, Inc. v. Bunner, 75 P.3d 1, 27 (Cal. 2003), *modified*, 2003 Cal. LEXIS 7676 (Cal. 2003).

25. *See generally* Alice Corp. Pty. v. CLS Bank Int'l, 573 U.S. 208 (2014) (curtailing the patent eligibility of computer-implemented inventions substantially).

26. *See In re* Gulack, 703 F.2d 1381, 1385 (Fed. Cir. 1983).

27. *In re* Russell, 48 F.2d 668, 669 (C.C.P.A. 1931).

28. Apple Inc. v. Motorola, Inc., 757 F.3d 1286, 1307–08 (Fed. Cir. 2014).

29. *See, e.g.*, *Windows API Index*, MICROSOFT (May 30, 2018), https://docs.microsoft.com/en-us/windows/desktop/apiindex/windows-api-list [https://perma.cc/7V4G-342X]; *Package Index*, ANDROID DEVELOPERS (June 6, 2018), https://developer.android.com/reference/packages.html [https://perma.cc/6THV-N3B4].

disclosure, it would be difficult for a platform owner to argue that their API is *secret*.

Some APIs are not published but are used internally in publicly available software. For example, a video game cartridge may communicate with a console using an API. Developers seeking to design a custom cartridge may reverse engineer the video game's API in two ways: disassembly and black box reverse engineering. In "disassembly," the developers examine the game's executable binary code to determine how the game communicates with the API.[30] In "black box" reverse engineering, the developers determine how the game communicates with the API by observing the behavior of the game (e.g., memory writes and reads) as it is being played.[31] With enough time and resources, both methods allow a motivated developer to derive the API. They can then use the API to program custom cartridges. This is essentially how Accolade bypassed Sega's potential trade secret claims in *Sega v. Accolade*.[32]

API creators can attempt to prevent this problem by licensing their software under a clause prohibiting reverse engineering.[33] This has been met with varying degrees of approval by the courts. Although the Federal Circuit (applying First Circuit law) has enforced a shrinkwrap license that prohibits reverse engineering,[34] the Fifth Circuit has held that any prohibition on reverse engineering is preempted by 28 U.S.C. § 117, which provides various rights for computer program copy owners.[35] Given this circuit split, the ability for trade secret law to protect publicly available APIs is far from guaranteed.

On the other hand, an API is clearly protectable as a trade secret if it is defined, implemented, and used entirely internally to an organization.[36] In such circumstances, there is no issue of reverse engineering or secrecy, because anyone authorized to access the API would be under a duty of secrecy to the organization.[37] However, the risk of appropriation for these kinds of APIs is limited because there is likely little commercial value in an API that is not used by anyone outside of the organization.

---

30. *See* JOHNATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL 2.0 4 (2011).
31. *Id.*
32. *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514–15 (9th Cir. 1992).
33. BAND, *supra* note 30, at 121.
34. Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1323 (Fed. Cir. 2003).
35. Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 270 (5th Cir. 1988).
36. *See* Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 475 (1974) (explaining that the element of secrecy required for trade secret protection "is not lost, however, if the holder of the trade secret reveals the trade secret to another 'in confidence, and under an implied obligation not to use or disclose it.'"); MELVIN F. JAGER, TRADE SECRETS LAW § 3:17 (2018).
37. *Kewanee*, 416 U.S. at 475; JAGER, *supra* note 36 ("In general, an employee owes a duty of loyalty to the employer with respect to all conduct within the scope of the employment. This loyalty obligation includes . . . the duty to refrain from using or disclosing confidential information acquired during employment." (citations omitted)).

3. Copyrighting APIs

In view of the deficiencies in patent and trade secret protection of APIs described above, it falls to copyright law to provide robust intellectual property protection for APIs. Copyright is the most intuitive IP right for APIs for several reasons. First, computer programs are copyright-eligible,[38] and APIs are often embodied in computer programs. Second, the substance of an API — the declaring code and the SSO — is suited for copyright for the same reason it is unsuited for patentability: it is essentially information fixed in a tangible medium.[39] Third, the work of creating an API — writing the declaring code and designing the SSO — is undoubtedly an "original work of authorship"; it involves creative choices that have nothing to do with functional considerations.[40] The two biggest challenges to API copyright, however, are the merger and fair use doctrines. Both doctrines have been applied to preclude copyright infringement claims where the purpose of copying was interoperability between two programs.

Under the merger doctrine, a work is not copyright eligible when its expression has merged with an idea.[41] In *Lotus v. Borland*, Borland released a spreadsheet program, Quattro, that used the same menu structure as Lotus's spreadsheet program, Lotus 1-2-3.[42] Borland copied Lotus's menu structure so that Quattro could run user macros (i.e., programs) written for Lotus 1-2-3.[43] The First Circuit held that the menu structure of Lotus 1-2-3 was an uncopyrightable method of operation (at least to the extent of Borland's use), in part because copying the menu structure was the only way Quattro could be functionally interoperable with Lotus 1-2-3.[44] On the other hand, the Federal Circuit in *Oracle v. Google* found that the Java API was copyrightable.[45] In that case, Google had copied the declaring code and SSO of Java, but did not release it in a way that was functionally interoperable with Sun/Oracle's Java.[46] Although *Oracle* was decided under Ninth Circuit law (and thus did not consider *Lotus* precedential), one can square these decisions by

38. *See* U.S. COPYRIGHT OFFICE, COPYRIGHT REGISTRATION OF COMPUTER PROGRAMS 1 (Sept. 2017), https://www.copyright.gov/circs/circ61.pdf [https://perma.cc/4K9P-9Q5B].

39. *See* 17 U.S.C. § 102(a) (2018) ("Copyright protection subsists . . . in original works of authorship *fixed in any tangible medium* of expression . . . ." (emphasis added)).

40. *See id.*

41. 17 U.S.C. § 102(b).

42. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 809–10 (1st Cir. 1995), *aff'd*, 516 U.S. 233 (1996). A menu structure is essentially an API because, like other APIs, it provides an interface through which a program can control external functionality.

43. *Id.*

44. *Id.* at 815.

45. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014).

46. *Id.* at 1351.

considering functional interoperability essential to the merger doctrine: an API is only uncopyrightable to the extent that copyright would prevent functional interoperability between programs.

The fair use doctrine under Ninth Circuit law considers functional interoperability similarly important. For example, *Sega v. Accolade* held that Accolade's unauthorized copying of Sega's code was nevertheless fair because Accolade's purpose was to "discover the *interface specifications* for the Genesis console" in order to understand "the functional requirements" for Genesis compatibility.[47] Similarly, *Sony v. Connectix* held that Connectix's copying of Sony's BIOS was fair use because Connectix's aim was to "observe the signals sent between the BIOS and other programs on the computer" (i.e., to understand the BIOS's interface).[48] This would allow Connectix to develop its own, functionally compatible software.[49] In *Oracle II*, the Federal Circuit, interpreting Ninth Circuit law, reversed the jury's finding of fair use, holding that Google's interoperability arguments were not of the same nature as those of the defendants in *Sega* and *Sony*.[50] In particular, the court noted that "even the 'modest' level of transformation at issue in *Sony* was more transformative than what Google did: copy code verbatim to attract programmers to Google's 'new and *incompatible* platform.'"[51]

In sum, despite the fair use and merger doctrines, copyright remains the best way to protect APIs.

III. THE RIGHT TO EXCLUDE OVERCOMPENSATES API OWNERS

This Part examines the *value* of API copyrights and concludes that copyright overcompensates API owners in one key dimension — namely, the ability of right holders to appropriate user switching costs (the "standardization value" of the APIs). This Part argues that preventing API copyright owners from appropriating standardization value averts harms to general welfare (at least to the extent that they are greater than for other copyrighted works), and does not disturb the incentive structure underlying copyright.

Section III.A discusses prior scholarship, which has identified several negative externalities that result from intellectual property protection of APIs. Section III.B divides the value of an IP right on an API into two discrete components: (1) the "expression value" of the API and (2) the "standardization value" of the API. Section III.C describes in further

47. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525–26 (9th Cir. 1992) (emphasis added), *amended* Jan. 6, 1993.

48. Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 599–600 (9th Cir. 2000).

49. *Id.* at 600.

50. Oracle Am., Inc. v. Google LLC (*Oracle II*), 886 F.3d 1179, 1210 (Fed. Cir. 2018).

51. *Id.* at 1200 (emphasis added).

detail the ways in which switching costs create standardization value. Section III.D posits that the negative externalities identified in prior scholarship are caused not due to IP protection of APIs per se, but instead because an unqualified right to exclude allows API owners to appropriate standardization value. Section III.D argues that an API's standardization value need not be appropriable for there to be sufficient incentive to create APIs. Section III.D further argues that preventing appropriation of standardization value, while leaving copyright protection of APIs otherwise intact, does not disturb API owners' "prospects" for desirable post-creation activity.

A. Prior Scholarship

Many commentators have pointed out the potentially detrimental results of intellectual property protection of software interfaces, such as APIs.[52] Underlying this conclusion is a longstanding recognition of network effects and switching costs inherent to software interfaces.[53] As Professor Pamela Samuelson points out, the value of an interface to each user increases with the number of developers and consumers using that interface.[54] Adoption of an interface by some developers and consumers makes it easier for the interface to attract new developers and consumers.[55] Thus, over time, a small and possibly arbitrary market advantage can result in a "positive feedback effect" that causes de facto monopolization — even where there are many other interfaces that are technically equivalent.[56] Commentators have also identified several types of negative externalities that can result from intellectual property rights over a standard interface.

First, competitors, in attempting to establish their interface as the de facto standard, may be incentivized to act in a manner contrary to social

52. *See e.g.*, Pamela Samuelson, *Are Patents on Interfaces Impeding Interoperability?*, 93 MINN. L. REV. 1943, 1964–65 (2009) [hereinafter Samuelson, *Patents on Interfaces*] (extensively discussing proposed and existing intellectual property rights regimes for software interfaces); Daniel Lin, *Research Versus Development: Patent Pooling, Innovation and Standardization in the Software Industry*, 1 J. MARSHALL REV. INTELL. PROP. L. 274, 279–81 (2002); 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. 305, 455–64 (2018).

53. *See* Brief Amicus Curiae of Economics Professors and Scholars in Support of Respondent, Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1995) (No. 94-2003), 1995 WL 728562, at *6–10; Samuelson, *Patents on Interfaces*, *supra* note 52, at 1951.

54. *See* Pamela Samuelson & Suzanne Scotchmer, *The Law and Economics of Reverse Engineering*, 111 YALE L.J. 1575, 1618 (2002).

55. *See id.*

56. *See* Lin, *supra* note 52, at 280–82; *see also* Joseph Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35, 46 (1989) (discussing the effect of network effects on intellectual property policy in general).

welfare.[57] For instance, standard owners may engage in "penetration pricing": pricing the standard below cost to spur adoption.[58] This practice distorts market incentives and may lead to an inferior interface being adopted simply because its owner could bear a loss longer than the competition. In addition, where a government or other body influences adoption of a standard, an interface owner may be incentivized to lobby the decision maker to mandate use of their standard.[59] Where there are multiple lobbying parties, each with a satisfactory interface, these lobbying expenditures have the potential to fully consume any social benefit provided by the interfaces.[60]

Second, intellectual property on interfaces can increase costs to interface users beyond the social value that the interfaces create. If an interface emerges as a market standard, its owner will be able to set monopolistic, rather than competitive, prices.[61] Even if a market has multiple alternative interfaces, interface owners can price their interfaces to take advantage of their users' collective inertia, switching costs, and other network lock-in effects.[62] These higher prices will exclude consumers from the market, creating a social deadweight loss.[63]

Third, granting IP rights to interfaces can have a detrimental effect on further innovation. For example, imagine the following scenario: (1) all users currently use Graphing Software A, which runs user macros written for API A; (2) a newcomer to the market releases Graphing Software B, which runs user macros written for API B; (3) Graphing Software B provides better features such that the users are willing to pay $10 more for it than for Graphing Software A; and (4) the switching cost from API A to API B is $20 per user, of which $5 is the price of API B, and $15 is the cost of rewriting the user's macros for API B.[64] Even though Graphing Software B is superior to Graphing Software A (it provides $10 of utility for a price of $5), no users will switch over. Thus, the superior product will be unable to enter the market. Professors Joseph Farrell and Garth Saloner have generalized this result; they show that in the absence of perfect information and coordination, consumers may

57. *See* Lin, *supra* note 52, at 282.

58. *See id.* at 282 n.41.

59. *See* Pamela Samuelson, *Questioning Copyrights in Standards*, 48 B.C.L. REV. 193, 223 (2007) [hereinafter Samuelson, *Questioning Copyrights*].

60. *See* WILLIAM M. LANDES & RICHARD A. POSNER, THE ECONOMIC STRUCTURE OF INTELLECTUAL PROPERTY LAW 17–20 (2003) (discussing deadweight losses caused by duplicative development costs).

61. Lin, *supra* note 52, at 282.

62. Menell, *supra* note 52, at 456–59.

63. *Id.*

64. This scenario is inspired by the facts of *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807 (1st Cir. 1995).

unanimously favor a new technology but never actually make the switch because no single consumer is sufficiently motivated to start the "bandwagon" that will overcome current network effects.[65] Thus, IP protections for APIs can prevent innovators from entering the market.

B. The Right to Exclude Enables Appropriation of an API's Standardization Value

All of the negative externalities discussed above — overspending on marketing, increased costs to consumers, and heightened barriers to innovation — arise from the right to exclude. And yet the exclusionary right undergirds both dominant theories of intellectual property. Under the traditional "reward" theory, the right to exclude is the reward for creation.[66] Under Edmund Kitch's "prospect" theory, the right to exclude defines the prospect upon which the creator may develop beneficial post-invention value (commercialization, future innovation, etc.).[67] So what makes the negative externalities described above for IP on APIs any different from IP on other subject matter? This Section answers that question by dividing the value of an API into two components: expression value and standardization value. The right to exclude from an API, unlike the right to exclude from any other copyright-eligible work, has standardization value. As will be discussed in further detail below,[68] an API owner's attempt to create and later appropriate its standardization value can cause negative externalities to such an extent that they overwhelm any benefit to social value incentivized by the right to exclude.

The "expression value" of a protected work is the amount of value provided by the work that can be appropriated because of the right to exclude. For example, the expression value of an action movie includes the excitement and suspense it generates in the watcher. The expression value of a textbook includes the clarity with which it explains a difficult subject. The expression value of a computer program (i.e., a software implementation) includes the value that comes from what the program does. Importantly, the expression value of a work can be increased by

---

65. *See* Joseph Farrell & Garth Saloner, *Standardization, Compatibility, and Innovation*, 16 RAND J. ECON. 70, 78 (1985). Farrell and Saloner's analysis is slightly different from the example discussed above because they assume that the benefit of the new product derives in part from whether others are using it (i.e., network effects). The example above illustrates the same point but with user switching costs (which can be — but need not be — due to network effects). For many real-world APIs, network effects and switching costs play distinct and important roles in maintaining market dominance.

66. Henry E. Smith, *Intellectual Property as Property: Delineating Entitlements in Information*, 116 YALE L.J. 1742, 1814–18 (2007).

67. *See* Edmund W. Kitch, *The Nature and Function of the Patent System*, 20 J.L. & ECON. 265, 266 (1977).

68. *See infra* Section III.D.

post-creation commercialization activities. Marketing, for instance, can increase a protected work's popularity. This, in turn, can increase the work's expression value. For example, part of the expression value of the latest Game of Thrones episode is being able to talk about it with friends and acquaintances who have also seen it. The price that a rights holder can charge for a protected work includes the expression value of that work.

APIs, like any other type of copyrighted work, can have substantial expression value. An API's expression value may arise, for example, from the API having a particularly elegant design or a large number of features, or from the amount of time and resources it took to design the API. The API's expression value can also be increased by a variety of post-creation activities. For example, an API's expression value can increase if its creator releases very extensive and easy-to-understand documentation for the API.

On the other hand, the "standardization value" of a protected work is the cost to switch to an alternative to the protected work that can be appropriated because of the right to exclude. The standardization value of an API includes the cost to developers who use an API to redevelop their existing applications for another API, and also includes the cost to end users (e.g., users of the applications developed by the third-party developers) to switch to a platform that uses another API. The price a rights holder can charge for a work also includes its standardization value.

It is important to note what standardization value does *not* include. It does not include transaction costs that the rights holder cannot appropriate, switching costs that would still exist even without the right to exclude, or the value of network externalities per se. Thus, standardization value would not include: (1) the cost of a new consumer deciding which of many alternative APIs to use, (2) the expense of cancelling a business relationship with one platform and starting one with another, or (3) the value to a new end user of an API with millions of other users.

There is a reason for this relatively narrow definition: The other types of transaction costs, switching costs, and network effects all exist in other subject matter covered by copyright. Virtually every copyright creates transaction costs that the rights holder cannot appropriate.[69] Many copyrighted works create switching costs that would still exist even without the right to exclude.[70] And many copyrighted works derive their value in part from network effects.[71]

---

69. One example of this is the cost of obtaining a copyright license.

70. For example, a book publisher would incur costs in switching its presses to a different book even if had a license to print both books.

71. The *Game of Thrones* example discussed above is just one example of this; part of the copyright's value is the community around the show.

APIs are the only copyright-eligible works for which the right to exclude, in itself, can create substantial switching costs. In other words, standardization value, as defined above, is unique to APIs. This is because a third-party reimplementation of an API (though prevented by the right to exclude) would essentially eliminate developer and end-user switching costs. The following Section discusses in further detail the components that make up an API's standardization value.

C. API-Related Switching Costs as the Origin of Standardization Value

This Section discusses the ways in which an API owner can appropriate value from the right to exclude. It does this by introducing the concepts of "API heterogeneity" and "API homogeneity." The right to exclude allows an API owner to maintain API heterogeneity in a market. API heterogeneity, in turn, allows the API's owner to appropriate its standardization value from two sources: (1) the costs for API developers to develop their software for a new API; and (2) the costs for API end-users to switch to a new API.

A software market has "API heterogeneity" if each software platform within a market uses a different API. Figure 3 shows an example of a software market with API heterogeneity. The desktop operating system market is a classic example of API heterogeneity: each of the major operating systems (Windows, macOS, and Linux) cannot run applications written for any other operating system.[72] Markets with API heterogeneity have high switching costs; for example, it would be expensive for a Windows user that owns several Windows-only applications to switch to macOS. In contrast, a market with "API homogeneity" is one where each software platform implements the same API. Figure 4 shows an example of a software market with API homogeneity. The modern web browser market has API homogeneity; all browsers today implement essentially the same API for web pages (which includes HTML, JavaScript, and CSS). In other words, a web page written for any specific web browser will run on any other web browser. Markets with API homogeneity have low switching costs; for example, an average Google Chrome user can switch to Mozilla Firefox at virtually no cost.[73]

---

72. This is not strictly true (see the Wine project on Linux, macOS's limited POSIX compatibility with Linux, and macOS's Parallels software), but I ignore these applications for the sake of simplicity.

73. This example is not perfect because Chrome and Firefox use different browser extension and add-on APIs. But, incidentally, many add-ons are compatible with multiple browsers because their browser add-on APIs tend to be very similar.

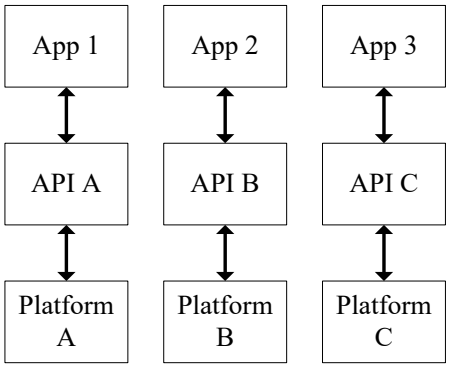

Figure 3: A Market with API Heterogeneity

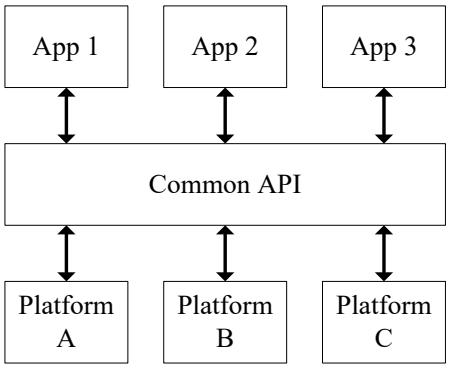

Figure 4: A Market with API Homogeneity

API heterogeneity is maintained by an API owner's right to exclude. This is because a self-interested owner of an API with substantial standardization value would either refuse to license it (thus allowing the owner to appropriate the standardization value of the API from the API's users, as discussed below), or only license the API for an exorbitant amount (i.e., by pricing the API's standardization value into the license fee). Both of these positions push new entrants to independently develop their own, competing APIs. This results in API heterogeneity.

API heterogeneity allows an API owner to increase prices on its users. The extent to which the owner can increase prices depends on the switching costs (1) for third-party API developers, and (2) for end users. Because these costs are only appropriable because of the API owner's right to exclude, they contribute to the API's standardization value.

An API's standardization value comprises switching costs for third-party application developers because API owners can raise prices in proportion to a developer's difficulty in switching APIs. APIs often vary substantially even when serving essentially the same purpose.[74] Thus, porting software written for a first API to a second API can entail considerable expense. For example, the developer needs to learn the second API or needs to hire someone who already knows it. In addition, for the second API, the developer may need to acquire new software development tools, which may also cost money. Furthermore, depending on the differences between the APIs, porting the application may require essentially rewriting the application's source code, a substantial cost. An API owner that understands these costs may increase the price charged to the developer to be above the price of competitors, but just below the price at which the developer would switch despite the switching costs. Thus, the API owner can essentially appropriate the developer switching costs.

An API's standardization value further comprises switching costs for end users. Some applications will be exclusive to a platform because their developers do not want to undertake the expense of reimplementing them using other APIs. Users of a platform will naturally use some of these applications. However, by using platform-exclusive applications, end users set themselves up for substantial switching costs. The most obvious switching cost is the actual cost of purchasing equivalent applications on the new platform. For example, a consumer with an iPhone may purchase a $5 iOS-only fitness app. If the consumer later wished to switch to a Samsung Galaxy phone, they would have to purchase an equivalent Android app (which might also cost $5). Switching costs also result from user familiarity with applications. For example, the iPhone user switching to a Samsung phone would need to spend the time to learn how to use the Android fitness app for the same purpose for which they used the iOS app. In addition, switching costs can result from platform-specific data. For example, the user may have data regarding prior workouts on their iOS fitness app that may be difficult or impossible to transfer to the equivalent Android app. An API owner that understands these costs may increase the price charged to the API's end users to be above the price of competitors, but just below the price at which the user would switch despite the switching costs. Thus, the API owner can also essentially appropriate end user switching costs.

This does not need to be the case. In a regime where the owner of an API cannot appropriate its standardization value, new entrants would be able to enter the market by licensing the existing API for a fair price, leading to API homogeneity. This would enable the market to enjoy the

74. For example, in many software markets, competitor APIs are written in different programming languages.

efficiencies of a natural monopoly without a natural monopoly's anti-competitive effects.

D. The Standardization-Value Appropriation Problem

This Section discusses the problem of standardization-value appropriation. It argues that all three social costs discussed at the start of Part III — overspending on marketing, increased costs to consumers, and higher barriers to innovation — are significantly amplified by an API owner's ability to appropriate standardization value. Although all three of these negative externalities are present to some extent with any grant of copyright, when an API owner is allowed to appropriate standardization value, these social costs can actually exceed the value of the API itself. This Section argues that preventing an API owner from appropriating standardization value (e.g., by forcing an API's owner to license the API for its expression value) will bring these social costs back in line with those of other subject matter protected by copyright. This Section separately examines the two dominant theories of intellectual property protection — the "reward" theory and the "prospect" theory — and concludes that preventing an API owner from appropriating standardization value does not thwart the incentive to innovate under either theory.

1. The Appropriation of Standardization Value Has Social Costs

Marketing expenses, in themselves, are not undesirable social costs. In fact, the commercialization theory of intellectual property considers them as expenditures that IP law ought to incentivize.[75] For most copyrightable works, marketing costs are limited to the expression value of the protected work; it would not make sense for a party to spend more because they would not be able to appropriate any more value from the copyright. However, an API's standardization value can be substantially more than its expression value.[76] Thus, when multiple firms are lobbying for their API to become a governmental or industry standard, they would each be willing to spend far more than the social value of the API. These duplicated lobbying costs increase deadweight losses to an extent greater than if the only value "at stake" for each API owner was its API's expression value.

Preventing an API owner from appropriating standardization value would also lower consumer deadweight losses. As discussed in detail above, an API's owner can charge higher prices by exploiting developer and end-user switching costs. However, if an API's owner was forced to

75. *See* Ted Sichelman, *Commercializing Patents*, 62 STAN. L. REV. 341, 366–67 (2010).
76. *See supra* Section III.C.

license it to a competitor for the API's expression value, end users and developers would be able to switch to the competitor for a lower cost. Accordingly, the API's owner could no longer incorporate switching costs into the price of the API, thus reducing the API's price and overall deadweight losses.

Barriers to innovation would also be reduced by preventing an API owner from appropriating standardization value. Recall the scenario discussed earlier in Part III.[77]

Now let us assume: (5) the expression value of API A is $1 per user; and (6) it costs $2, amortized per user, for the developer of Graphing Software B to reimplement API A (so that users can run macros written for API A in Graphing Software B). Under a regime where API A must be licensed to the developer of Graphing Software B for its expression value, the developer of Graphing Software B will reimplement API A to eliminate its potential customers' switching costs. The reimplementation increases the price of Graphing Software B to $8: $5 to develop the software, $1 to license API A, and $2 to reimplement API A. In this regime, Graphing Software B will be able to enter the market because users can switch to it without having to rewrite their macros; users will be willing to spend $8 on the price of the software to gain $10 of utility. Thus, preventing standardization-value appropriation reduces barriers to innovation.

2. API Owners Need Not Appropriate Standardization Value To Be "Rewarded" for Their Creation

One could argue that the right to exclude — and the concomitant ability for an API owner to appropriate standardization value — is necessary to provide the optimal incentive to API creators.[78] In particular, because of the network effects inherent to APIs, competitors in a regime without standardization-value appropriation may adopt a "wait and see" approach. Instead of creating an API (which may involve considerable

---

77. In the scenario: (1) all users currently use Graphing Software A, which runs user macros written for API A; (2) a newcomer to the market releases Graphing Software B, which runs user macros written for API B; (3) Graphing Software B provides better features such that the users are willing to pay $10 more for it than for Graphing Software A; and (4) the switching cost from API A to API B is $20 per user, of which $5 is the price of API B, and $15 is the cost of rewriting the user's macros for API B. As noted earlier, Graphing Software B will be unable to enter the market if the developer of Graphing Software A is allowed to appropriate API A's standardization value (i.e., the $15 switching costs), because the developer of Graphing Software A will refuse to license API A to Graphing Software B. *See supra* Section III.A.

78. *Cf.* William M. Landes & Richard A. Posner, *An Economic Analysis of Copyright Law*, 18 J. LEGAL STUD. 325–26 (1989) (describing copyright as a tradeoff between "the costs of limiting access to a work" and "the benefits of providing incentives to create the work in the first place").

expense and has no guarantee of being successful in the market), they will wait to license the API that is ultimately adopted by the market. This approach could delay or even preclude the development of APIs. Yet there are several reasons to believe that any API creation would be sufficiently incentivized even without the creator's ability to appropriate standardization value.

First, people build APIs because they have to and would do so even if they were not provided any exclusionary right over them. People create computer programs for many reasons, including the fact that they can appropriate the expression value of computer software via copyright. In order to create computer programs that communicate with each other, people need to design and implement APIs. So, the ability to appropriate the value of a computer program that uses an API may be sufficient reward to incentivize the creation of the API. The history of API copyright eligibility seems to support this view: APIs were uncopyrightable for many years,[79] yet those years saw a great deal of API development.

Second, to the extent that API creation needs to be "rewarded" separately from computer programs, API creators can be sufficiently incentivized by several other appropriation mechanisms. For example, API creators can appropriate an API's value via first-mover advantage and the difficulty of reverse engineering the API. In addition, a legal regime may require those who copy an API to pay the API's creator a license fee commensurate with its expression value.[80]

Third, the chance to appropriate standardization value may have little motivational impact. Corporations, especially those large enough to have a high chance of API adoption, tend to be risk-averse.[81] That is, they prefer lower-risk investments, even if the expected value return on the low-risk investment is lower than that of a high-risk investment. Risk-averse corporations are unlikely to be motivated by standardization value because, although the potential rewards are great, there is high uncertainty as to which players, of all participants in the market, will ultimately become one of the few market leaders with standardization value.

Fourth, assuming that software is copyrightable, an API cannot be commercialized by a third party without that party also reimplementing

---

79. *See* Samuelson, *supra* note 18, at 13.

80. *See* Ted Sichelman & Stuart J.H. Graham, *Patenting by Entrepreneurs: An Empirical Study*, 17 MICH. TELECOMM. & TECH. L. REV. 111, 118 (2010).

81. *See* George Deeb, *The 5 Reasons Big Companies Struggle with Innovation*, FORBES (Jan. 8, 2014), https://www.forbes.com/sites/georgedeeb/2014/01/08/the-five-reasons-big-companies-struggle-with-innovation/#7f62685e2958 [https://perma.cc/74MT-SG3Y]; Ben Casselman, *Risk-Averse Culture Infects U.S. Workers, Entrepreneurs*, WALL STREET J. (June 2, 2013), http://www.wsj.com/articles/SB10001424127887324031404578481162903760052 (last visited Dec. 20, 2018).

the API. Reimplementing an API can require large amounts of time and money, which reduces the risk of free riding.[82]

Fifth, assuming the API's owner continues to hold rights to derivative works, only the API's owner can improve the API. Thus, an API licensee would be perpetually playing "catch up" to the API owner's implementation, and the licensee would not be able to add any value to the API over the API owner.

In sum, innovation is likely to occur whether or not API owners are permitted to appropriate standardization value in order to incentivize innovation in APIs and allowing for standardization-value appropriation may not substantially increase innovation.

3. Preventing Appropriation of Standardization Value Does Not Disturb the API Owner's "Prospect" for Desirable Post-Creation Activity

Under Edmund Kitch's "prospect" theory, the right to exclude defines the prospect upon which the creator may develop beneficial post-invention value.[83] Beneficial post-invention value can include "any activity following the initial invention that leads to a commercially available product or service — including developing, testing, manufacturing, sales, and service of the initial invention, as well as the invention and subsequent development of improvements . . . ."[84] Under this view, preventing the API owner from appropriating standardization value — thus voiding the right to exclude — reduces the efficiency by which post-creation value will be developed.

For example, one could argue that preventing appropriation of standardization value reduces the creator's incentive to market the API.[85] In turn, consumers are less informed than they otherwise would have been. But marketing that aims to better inform developers and consumers of the benefits of an API (e.g., documentation) would still be incentivized because such marketing would increase the expression value of the platform (i.e., the utility the users derive from the API). On the other hand, marketing would not be incentivized where it does not increase the expression value of the API. Both effects are socially desirable. As noted at the start of Part III, allowing an API owner to appropriate standardization value would create perverse incentives for

---

82. *See* LANDES & POSNER, *supra* note 60 ("[W]hen costs of duplication are high, free riding may be eliminated, and intellectual property protection may therefore become relatively unimportant.").

83. Kitch, *supra* note 67.

84. Sichelman, *supra* note 75, at 354.

85. *Cf.* Landes, *supra* note 78, at 326.

API owners to invest excessive amounts of money persuading governments to adopt their API as a standard.

One also could argue that the right to exclude provides an important incentive to commercialize an API. In particular, the right to exclude may be necessary to incentivize creation of software that implements that API. Yet, there is reason to doubt this view. Most importantly, implementing software is separately protectable by copyright.[86] Thus, the law already prevents a free-rider from copying the implementing software without compensating its creator. Without this risk of free riding, there appears to be no reason to provide the implementing software with additional IP protection.[87]

One also might contend that the right to exclude gives an API creator much-needed breathing room for further innovation. In particular, the Kitchian view is that an API creator is in the best position to conduct and/or coordinate follow-on innovation. However, preventing standardization-value appropriation does not disturb this view because it does not require abrogating an API copyright holder's derivative works right. For example, one way to prevent standardization-value appropriation is with a regime that provides for copyright but also compulsory licensing, of APIs.[88] In such a regime, competitors may license an API for a fair royalty but cannot make any changes. Any improvements or other changes to the API would have to be made by the owner, and the owner could appropriate the value of these changes via its own profits and via royalties from its licensees. Thus, the owner would remain in control of, and the beneficiary of, follow-on innovation.

In sum, an API's owner does not need to appropriate the API's standardization value to be incentivized to commercialize the API.

IV. POSSIBLE SOLUTIONS TO THE STANDARDIZATION-VALUE APPROPRIATION PROBLEM

If we accept that an API owner should not be able to appropriate its standardization value, the next task is to determine what the scope of legal protection for APIs should be. Should APIs be uncopyrightable altogether? Should an API user have a fair use defense if their use of an API is predominantly for its standardization value, and not for its expression value? Should there be a compulsory licensing regime for APIs,

86. 17 U.S.C. § 102 (2018).

87. *See* Landes, *supra* note 78, at 370 ("[I]f one believes that unprotected ex post commercialization activity does not lead to pernicious free riding by others . . . then these reactions [i.e., the narrowing of the scope of IP rights] largely seem sensible.").

88. *See infra* Section IV.C.

similar to the existing regime for songs? This Part concludes that all of these approaches have significant problems.

A. Should APIs Be Uncopyrightable?

One tempting solution to the problem of standardization-value appropriation is to simply declare APIs to be uncopyrightable subject matter. This type of bright-line rule would have the advantage of certainty; actors could easily tell whether their behavior was permissible or not. Such a rule also would have low enforcement costs: a lawsuit alleging copyright infringement of an API likely would be dismissed at the pleadings stage. However, a strict rule banning copyright on APIs would be undesirable for several reasons.

First, there is no conceptual reason why API creators should be prevented from appropriating the expression value of what is undoubtedly a creative work. The work of designing an API — choosing method and class names, writing declaring code, and arranging the SSO of classes, packages, etc. — is clearly a creative endeavor.[89] Copyright incentivizes creators in many other fields — art, literature, filmmaking, etc. — to appropriate the expression value of their creations via copyright. To deny API creators the same right is arbitrary and ultimately risks disincentivizing API creation relative to the creation of other works. Without API copyrights, market participants would be even more likely to take a "wait and see" approach of waiting until an API gained traction and free riding on that API. Where multiple market participants take this approach, development of APIs could be delayed or even be precluded.

Second, making APIs uncopyrightable would disincentivize post-creation development of an API's value. Much of the utility of many APIs arises not from the definition of the API itself but from complementary products that make using the API easier and more effective: clear documentation, intuitive tutorials, powerful development tools, etc. These products may, to some extent, be separately protectable by IP. But even if the documentation, tutorials, and other complementary products are protected, they can still create "positive spillover" effects such that competitors who use the API also benefit from those products.[90] Thus, without copyright, an API creator (or others) would be less likely to invest in increasing the API's value, even if it would be socially desirable.

---

89. *See supra* Section II.A; Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1361 n.6 (Fed. Cir. 2014) ("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 . . . . [I]t took years to write some of the Java . . . .").

90. *Cf.* Sichelman, *supra* note 75, at 370.

Third, a complete lack of rules on the use of APIs could lead to opportunism whereby a new entrant can abuse their power in other markets to disadvantage their competitors. The archetypal example of this was Microsoft Corporation's strategy of "Embrace, Extend, and Extinguish" in the 1990s.[91] The strategy involved Microsoft adopting, or "embracing," an existing standard API and tapping into the existing market. Then, once Microsoft had sufficient market share (which might be achieved by bundling, as in the case of Internet Explorer), it would "extend" the API in non-standard ways that its competitors did not support. Thus, the Microsoft product would provide a superior user experience. Finally, over time, network effects would cause competitors to lose their market share and to fade into obscurity.[92] In essence, Microsoft would disadvantage its competitors by making subtle and undocumented modifications to the open standard.[93]

Although such practices could be actionable under antitrust law ex post,[94] copyright's derivative work right allows API owners to stop these types of actions ex ante by preventing competitors from releasing proprietary extensions.[95] For example, Adobe famously refused to license the PDF standard to Microsoft for fear that Microsoft would "embrace and extend" it.[96]

B. Should Use of an API for Interoperability Be Fair Use?

The courts' current answer to the standardization-value appropriation problem appears to be the fair use doctrine. Specifically, the courts in *Sega*, *Sony*, and *Oracle* appear to inject interoperability into the "purpose and character of use" prong of the fair use doctrine.[97] If the purpose of an API's use is functional interoperability, then the use is judged to be fair.[98]

All three of these decisions are consonant with the policy argument put forth in Part III of this Note: an API owner should not be permitted

---

91. *Embrace, Extend, and Extinguish*, WIKIPEDIA, https://en.wikipedia.org/wiki/Embrace,_extend,_and_extinguish [https://perma.cc/4TQW-KE3A].

92. *See U.S. v. Microsoft: Proposed Findings of Fact*, U.S. DEP'T. OF JUSTICE (2015), https://www.justice.gov/atr/us-v-microsoft-proposed-findings-fact [https://perma.cc/L6GT-Y4JY].

93. *See id.*

94. *See id.*

95. *See* 17 U.S.C. § 106(2) (2018).

96. CIO Staff, *Adobe Speaks Out on Microsoft PDF Battle*, CIO (Jun. 14, 2006), http://www.cio.com/article/2446013/compliance/adobe-speaks-out-on-microsoft-pdf-battle.html [https://perma.cc/XT6W-JTEW].

97. *See* discussion *supra* Section II.B.3.

98. *See id.*

to appropriate their API's standardization value. In particular, competitor reimplementation of an API is desirable because it reduces switching costs for consumers. Allowing an API owner to exclude competitors from such reimplementation would keep those switching costs in place, allow the API owner to appropriate standardization value, and increase consumer deadweight losses. Thus, courts use the fair use doctrine to limit the API owner's right to exclude.

However, the binary nature of the fair use doctrine makes it a problematic mechanism for preventing standardization-value appropriation. An accused use is either infringing, and injunctive relief may be granted, or the use is fair, and there is no liability. This can be a less-than-optimal result because, in many cases, a competitor makes use of an API for both its standardization value and its expression value.

Consider, for example, the Java API. Sun, and later Oracle, have spent significant resources creating and documenting the Java API to make it one of the easiest programming languages to learn and use.[99] Yet, the district court in *Oracle* held on remand that a reasonable jury could have found that Google's use of the API was fair because it was necessary for, as one commentator put it, "human interoperability."[100] Human interoperability, where a person becomes "trained, experienced, accustomed to using [an API] in the course of developing new works," could be considered a component of standardization value. In sum, Google ostensibly has taken advantage of both Java's expression value and its standardization value.[101] A finding of fair use thus swings the pendulum from overcompensating API owners to undercompensating them and disturbs the incentives to both create and commercialize APIs.

C. Should APIs Be Subject to Fixed-Rate Statutory Licensing?

Yet another solution to the standardization-value appropriation problem is through a statutory licensing regime similar to that which exists for musical works under Section 115 of the Copyright Act.[102] In such a regime, APIs would be licensable for a fee calculated based on a base

---

99. Alan Henry, *Five Best Programming Languages for First-Time Learners*, LIFEHACKER (Jan. 5, 2014), http://lifehacker.com/five-best-programming-languages-for-first-time-learners-1494256243 [https://perma.cc/56D7-MWA3]; Kavita Iyer, *Five Simple Coding Languages to Learn for First-Time Learners*, TECHWORM (Mar. 31, 2016), https://www.techworm.net/2016/03/5-simple-coding-languages-learn-first-time-learners.html [https://perma.cc/46HT-78SN].

100. *See* Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA, 2016 WL 3181206, at *6 (N.D. Cal. June 8, 2016); *see also* Johnathan Band, *Oracle v. Google and Interoperability*, 23, https://www.law.berkeley.edu/wp-content/uploads/2015/09/Jonathan-Band-oracle-vs-google-interoperability.pdf [https://perma.cc/7VAY-LY7Y].

101. *See* Band, *supra* note 100.

102. 17 U.S.C. § 115 (2018).

rate set ex ante by an administrative board like the Copyright Royalty Board. For example, the administrative board might set a per-user, per-device, or per-core rate on the use of an API. The API license fee would be that rate multiplied by the number of users, devices, or processor cores that use the licensee's API implementation. Alternatively, the royalty rate might be a certain percentage (e.g., 5%) of the total revenue of the software product that uses the API. Such a regime would ensure that API owners would be compensated for use of their API in a way that has relatively low delineation, abiding, and enforcement costs.[103]

The main problem with the fixed-rate statutory licensing regime described above is its inflexibility. APIs can vary substantially in size, complexity, and other factors that influence the investment needed to create one. In addition, some APIs may be far easier to commercialize than others. Thus, the optimal size of a "reward" or "prospect" is likely to vary significantly depending on the API. A fixed-rate licensing regime can take into account this variety in some limited ways. For example, if APIs that run on more processor cores require more investment to create, licensing on a per-core basis would be a relatively low-information-cost means for calculating an optimal royalty rate. However, any correlations between the optimal reward for innovation and low-information-cost variables like the number of users or processor cores are likely to be weak at best.[104] Thus, a fixed-rate statutory licensing regime is likely to undercompensate certain types of APIs and overcompensate others, leading to a lopsided incentive structure for innovation.

V. A VARIABLE-RATE COMPULSORY LICENSING REGIME FOR APIS

All three regimes discussed above in Part IV solve the standardization-value appropriation problem, but they all have the side effect of less-than-optimally incentivizing innovation. This Part proposes a variable-rate compulsory licensing regime for APIs.

Under this regime, an owner of an API with significant standardization value would be required to license use of the API for a fee that is commensurate with the API's expression value but does not consider the

---

103. *See generally* Robert P. Merges, *Compulsory Licensing vs. the Three "Golden Oldies" Property Rights, Contracts, and Markets*, 508 POL'Y ANALYSIS 1 (2004); *cf.* Henry E. Smith, *Property and Property Rules*, 79 N.Y.U. L. REV. 1719, 1719–31 (2004).

104. For example, the Fortran programming language, which has a relatively simple API, is the de facto standard for supercomputers, despite the existence of far more sophisticated languages. Lee Phillips, *Scientific Computing's Future: Can Any Coding Language Top a 1950s Behemoth?*, ARSTECHNICA (May 7, 2014) https://arstechnica.com/science/2014/05/scientific-computings-future-can-any-coding-language-top-a-1950s-behemoth [https://perma.cc/8G24-444C].

API's standardization value. In practice, the fee could be calculated in a similar manner to fees calculated for patent holders who have promised to license on fair, reasonable and nondiscriminatory ("FRAND") terms. Using FRAND-patent licensing fee cases as the baseline,[105] a variable rate-license scheme could operate as follows:

Identify the "smallest salable unit" that requires the use of the API.[106] For example, if the API at issue is that of a graphing application, the smallest saleable unit would be the graphing application itself — not the laptop or desktop computer that runs the application.

(1) Imagine a hypothetical arms-length negotiation between the API owner and the potential licensee at a time after the API is designed and commercialized but before the API has been adopted by any users.[107]

(2) Determine whether any alternative APIs existed at the time of the hypothetical negotiation, and the extent to which they could have been substituted for the API at issue. If the alternative APIs are proprietary they should not reduce the royalty rate as much as if they are in the public domain.[108]

Keeping in mind the smallest saleable unit, the time of the hypothetical negotiation, and the alternative APIs available at that time, determine the royalty rate that would have been arrived at in the hypothetical negotiation using the Georgia-Pacific factors.[109]

Under this regime, an injunction would only be awarded if an API user either refused to accept a royalty in accordance with the above principles or failed to pay a previously-negotiated royalty.

Determining a royalty rate as described above has several advantages. First, it allows the API owner to get a reasonable return on

---

105. Stan Lewis, *Valuing FRAND-Obligated Patents: An Emerging Consensus*, LAW360 (Nov. 27, 2013), https://www.law360.com/articles/487354/valuing-frand-obligated-patents-an-emerging-consensus (last visited Dec. 20, 2018).

106. *Cf.* LaserDynamics, Inc. v. Quanta Comput., Inc., 694 F.3d 51, 67 (Fed. Cir. 2012) ("[I]t is generally required that royalties be based not on the entire product, but instead on the 'smallest salable patent-practicing unit.'" (citation omitted)).

107. *Cf.* Ericsson, Inc. v. D-Link Sys., Inc., 773 F.3d 1201, 1232 (Fed. Cir. 2014) ("[T]he patentee's royalty must be premised on the value of the patented feature, not any value added by the standard's adoption of the patented technology.").

108. *Cf.* Microsoft Corp. v. Motorola, Inc., No. C10-1823JLR, 2013 WL 2111217, at *19 (W.D. Wash. Apr. 25, 2013) ("[T]he parties to a hypothetical negotiation under a RAND commitment would consider alternatives that could have been written into the standard instead of the patented technology. The focus is on the period before the standard was adopted and implemented . . . .").

109. *See* Georgia-Pacific Corp. v. U.S. Plywood Corp., 318 F. Supp. 1116, 1120 (S.D.N.Y. 1970). The *Georgia-Pacific* factors include the royalties obtained by the IP owner from the accused infringer or others, the duration of the license, the profitability of the IP, and the utility advantages of the IP over its alternatives. *Id.*

their creative work and subsequent commercialization efforts — ensuring continued incentives for API creation and beneficial post-creation activity.[110] Second, it allows the API owner the ability to control future changes to the API, preventing "extend, embrace, extinguish" types of abuse. Third, because all third-party API use would require a license fee, this regime would not suffer from the over- and under-compensation problems of a fair use regime or a compulsory fixed-rate licensing regime. Thus, the variable-rate compulsory licensing regime described above solves the standardization-value appropriation problem without suffering from the same side effects as uncopyrightability, fair use, and compulsory licensing regimes.[111]

However, this regime likely would have higher transaction costs than any other regimes described above. In particular, a royalty rate would have to be separately determined — possibly in court — for each API, or maybe even for each API licensee. But the API market may be able to bear these transaction costs better than industries that use fixed-rate schemes, like the music industry, for several reasons. First, the average value of a single use of an API is likely much higher than that of a single public performance of a song. In addition, a single song might be played by thousands of different entities such as restaurants, movie theaters, etc. Negotiating with each of these parties would be cost-prohibitive. The average API would likely be licensed to far fewer licensees.

In addition, there are ways to lower the transaction costs of such a regime. Professors Mark Lemley and Carl Shapiro haveproposed a FRAND licensing system whereby an API owner and a potential licensee attempt to negotiate a fee.[112] If that negotiation fails, the parties can move to "baseball-style" binding arbitration. In this arbitration style, each party submits a single offer to the arbitrator, who then picks one of those two offers. In eliminating costly rounds of argument, this process presents a relatively low-cost way to set a reasonably accurate royalty.

VI. CONCLUSION

This Note divides the value of an API copyright into two components: the API's "expression value" and its "standardization value." An API's standardization value is essentially the user switching costs that the right to exclude allows an API owner to appropriate. The Note notes that APIs are the only copyright-eligible works that allow their owner to

110. *See supra* Sections III.D.2, III.D.3.

111. *See supra* Part IV.

112. Mark A. Lemley & Carl Shapiro, *A Simple Approach to Setting Reasonable Royalties for Standard-Essential Patents*, 28 BERKELEY TECH. L.J. 1135 (2013).

appropriate standardization value, and thus copyright overcompensates API owners. The Note argues that appropriation of standardization value is unnecessary to incentivize innovation, and actually causes many of the social harms that commentators have long associated with copyright on APIs. The Note discusses four legal regimes that prevent an API's owner from appropriating its standardization value and concludes that a variable-rate compulsory licensing regime has manageable transaction costs and provides creators with the closest-to-optimal incentives for innovation.