2017

Software's Copyright Anticommons

Clark D. Asay
BYU Law, asayc@law.byu.edu

SOFTWARE'S COPYRIGHT ANTICOMMONS

Clark D. Asay[*]

Scholars have long assessed "anticommons" problems in creative and innovative environments. An anticommons develops when an asset has numerous rights holders, each of which has a right to prevent use of the asset, but none of which has a right to use the asset without authorization from the other rights holders. Hence, when any one of those rights holders uses its rights in ways that inhibit use of the common asset, an anticommons may result.

In the software world, scholars have long argued that anticommons problems arise, if at all, because of patent rights. Copyright, on the other hand, has not been viewed as a significant source of anticommons problems. But this Article argues that copyright is an increasingly significant cause of anticommons concerns in the software context for at least two related reasons. First, the increasingly collaborative nature of much modern software innovation means that any given software resource is subject to dozens, hundreds, or even thousands of distinct copyright interests, each of which can ultimately hamper use of the software resource. While collaborative innovation licensing models help reduce the threat of any given copyright holder restricting use of the software resource, these licensing models do not altogether eliminate such risks and, in fact, actually create risks of holdup and underuse that have previously received less attention than they are due. Second, interoperability needs in the growing "Internet of Things" and "cloud" economies demand sharing and reuse of software for these ecosystems to work. Yet because these technological ecosystems implicate thousands of different parties with distinct copyright interests in their software, the threat of any one of those parties ultimately using its rights in ways that inhibit the successful development and use of the Internet of Things and cloud economies looms large. In order to illustrate some of these anticommons

---

[*] Clark D. Asay. Associate Professor of Law, Brigham Young University Law School. Many thanks to Stephanie Bair, Julie Cohen, David Fagundes, Janet Freilich, Robert Gomulkiewicz, Patrick Goold, Paul Heald, Dmitry Karshtedt, Jake Linford, Jonathan Masur, Matthew Sag, Zahr Said, Andres Sawicki, Lisa Grow Sun, and participants at the 2016 Works-in-Progress in Intellectual Property Colloquium at the University of Washington, the 2016 Junior Scholars in Intellectual Property Workshop at Michigan State University, and a paper workshop at BYU Law, for helpful feedback on earlier versions of this Article.

problems in practice, this Article examines a recent high-profile software copyright dispute between Oracle and Google.

As a possible solution to these types of problems, this Article assesses the merits of more explicitly adapting copyright's fair use defense to the collaborative and interconnected nature of modern software innovation. The Article concludes by arguing that copyright disputes in other fields of creativity characterized by collaborative, interconnected development may also merit such fair use adaptations. Otherwise, anticommons problems may increasingly affect those fields as well.

INTRODUCTION

In 2012, Cindy Lee Garcia sued Google Inc. for copyright infringement.[1] Her lawsuit was meant to force the company to remove from YouTube an anti-Islamic film that included a five-second performance by her.[2] Garcia claimed a copyright interest in the performance and that YouTube, therefore, had no right to host it without her permission.[3] For $500, Garcia had agreed to the performance with the understanding that it would be used in a film called *Desert Warrior*.[4] But when her performance was distorted and used in an anti-Islamic film, *Innocence of Muslims*, the consequences were severe.[5] When the film appeared on YouTube and elsewhere, Garcia received death threats.[6] Some even suggested the attack on the U.S. embassy in Benghazi, Libya, was in response to the film.[7]

After the district court held against Garcia, the Ninth Circuit Court of Appeals initially ruled in her favor, holding that Garcia had likely met her burden of demonstrating a copyright interest in her performance.[8] This was so because, among other reasons, her performance included some amount of creativity, even if it was only five seconds long and based on a script provided

---

[1] Stefan Mentzer & Priya Srinivasan, *The* Garcia v. Google *Controversy and What It Means for Content Owners and Users*, LEXOLOGY (Mar. 20, 2014), http://www.lexology.com/library/detail.aspx?g=3b6ff140-49a1-4a85-9f90-b27d5987937a.

[2] *Id.*

[3] *Id.*

[4] Garcia v. Google, Inc., 766 F.3d 929, 932 (9th Cir. 2014).

[5] *Id.*

[6] *Id.*

[7] Scott Shane, *Clearing the Record About Benghazi*, N.Y. TIMES (Oct. 17, 2012), http://www.nytimes.com/2012/10/18/us/politics/questions-and-answers-on-the-benghazi-attack.html (highlighting that Susan Rice, then U.S. ambassador to the United Nations, suggested that the film helped cause the attacks).

[8] *Garcia*, 766 F.3d at 935–38, 940.

to her.[9] According to the court, that creativity may include her "body language, facial expression and reactions to other actors and elements of a scene."[10] But in early 2015, a full panel of the Ninth Circuit reversed this earlier decision, observing that "[t]reating every acting performance as an independent work [subject to copyright] would not only be a logistical and financial nightmare, it would turn [a] cast of thousands into a new mantra: copyright of thousands."[11]

Now fast forward to June 29, 2015. On that day, the U.S. Supreme Court denied Google's petition for writ of certiorari requesting review of the Court of Appeals for the Federal Circuit's decision in *Oracle v. Google*.[12] As a result, the Federal Circuit decision, which upheld copyright protection for certain parts of Oracle's Java software technologies, was left intact.[13] In particular, Google's use of thirty-seven of Oracle's Java "application programming interfaces" (APIs) in its Android operating system may constitute copyright infringement because, according to the decision, creation of the APIs required some creativity.[14] And this was so despite the fact that the APIs were, quantitatively, only a very small part of Android; Google engineers wrote nearly all ten million lines of the software code for Android.[15] Although the district court found on remand that Google's use of the APIs constituted fair use, the Federal Circuit's decision regarding the copyrightability of the APIs remained otherwise undisturbed.[16]

While these two cases have many obvious differences, they highlight a similar potential copyright problem: what the Ninth Circuit in *Garcia* called "copyright of thousands." More traditionally, this type of problem is referred to as an "anticommons" problem, which is shorthand for underuse of a resource because numerous parties have rights in the resource, and the presence of these

---

[9] *Id.* at 935.

[10] *Id.* at 934.

[11] Garcia v. Google, Inc., 786 F.3d 733, 742–43 (9th Cir. 2015).

[12] Lawrence Hurley & Dan Levine, *U.S. Top Court Declines to Hear Google Appeal in Oracle Java Fight*, REUTERS (June 29, 2015, 11:49 AM), http://www.reuters.com/article/usa-court-google-idINKCN0P91O S20150629.

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

[14] *Id.* at 1347, 1356.

[15] *Id.* at 1351 ("It is undisputed, however, that Google wrote its own implementing code, except with respect to: (1) the rangeCheck function, which consisted of nine lines of code; and (2) eight decompiled security files."); Timothy B. Lee, *Microsoft's Android Shakedown*, FORBES (July 7, 2011, 8:00 AM), http://www.forbes.com/sites/timothylee/2011/07/07/microsofts-android-shakedown/ (indicating that Android includes nearly ten million lines of software code).

[16] Mike Masnick, *Big Win for Fair Use: Jury Says Google's Use of Java API's Was Fair Use. . . On to the Appeal*, TECHDIRT (May 26, 2016, 2:01 PM), https://www.techdirt.com/articles/20160526/13584834558/ big-win-fair-use-jury-says-googles-use-java-apis-was-fair-use-to-appeal.shtml.

multiple rights inhibits others from using that resource in socially beneficial ways.[17] Scholars often argue that anticommons result when any one of these rights holders asserts its rights to prevent others from using the common resource.[18] But while rights assertions may be one common cause of anticommons problems, this Article takes the position that anticommons can also result when rights remain unasserted, or even, in some cases, when they are licensed. In other words, a multiplicity of rights in a resource can still result in underuse of that resource, even without formal bargaining breakdowns. This Article focuses on potential anticommons problems in the software context and argues that at least two significant and growing trends in modern software innovation are leading to rising anticommons concerns.

First, because increasingly more software products are collaboratively built by a variety of parties,[19] any given software product may be subject to hundreds, and sometimes even thousands, of copyright interests.[20] And each of these copyright holders may assert or use its rights in ways that make using the collaboratively built resource more difficult.[21] This difficulty may thus result in significant underuse of the software resource. The *Oracle v. Google* case, where Oracle successfully created copyright infringement concerns about Android on the basis of a copyright claim pertaining to a very small piece of the overall Android system, provides one recent example of this type of holdup problem. But even when the general intent of all parties involved in producing some software good is one of openness, the presence of numerous copyrights

---

[17] Michael A. Heller, *The Tragedy of the Anticommons: Property in the Transition from Marx to Markets*, 111 HARV. L. REV. 621, 673–79 (1998); Michael A. Heller & Rebecca S. Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, 280 SCI. 698, 698 (1998).

[18] Heller & Eisenberg, *supra* note 17.

[19] Carliss Baldwin & Eric von Hippel, *Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation*, 22 ORG. SCI. 1399, 1401 (2011) (arguing that collaborative models of open innovation are increasingly displacing "producer" models of innovation, in software and elsewhere); Charles C. Snow et al., *Organizing Continuous Product Development and Commercialization: The Collaborative Community of Firms Model*, 28 J. PRODUCT INNOVATION MGMT. 3, 8 (2011) (noting that the study of collaborative models of innovation has focused primarily on open source software communities, while arguing that these innovation models are increasingly applicable elsewhere as well).

[20] *See, e.g.*, *Torvalds/Linux*, GITHUB, https://github.com/torvalds/linux (last visited Sept. 20, 2016) (listing the number of contributors to Linux, the popular operating system, as the symbol for infinity).

[21] *See* AXEL BRUNS, BLOGS, WIKIPEDIA, SECOND LIFE, AND BEYOND: FROM PRODUCTION TO PRODUSAGE (2008) (arguing that the presence of expansive copyright rights in a variety of source materials can lead to "copyright thickets" that make use of such materials by follow-on creators more difficult, thereby resulting in an anticommons). *But see* Dan L. Burk, *The "Creating Around" Paradox*, 128 HARV. L. REV. F. 118 (2015) (noting the possibility of "copyright thickets" while doubting their actual existence).

increases costs for those wishing to use the collaboratively built good.[22] And these costs may slow use of the resource in ways that inhibit software innovation.[23]

Other scholars have assessed potential anticommons problems in the software context.[24] But they typically attribute any such problems to excessive patent rights, not excessive copyright rights.[25] Indeed, previous scholarship has often argued that copyright law has built-in safeguards against anticommons problems.[26] Furthermore, previous scholars have typically argued that collaborative innovation licensing models reduce, rather than produce, anticommons problems.[27] In fact, the primary issue scholars have assessed in the software copyright context is the proper boundary between copyrightable and non-copyrightable elements of software, a question that was very much at issue in the *Oracle v. Google* case.[28]

---

[22] *See* Clark D. Asay, *A Case for the Public Domain*, 74 OHIO ST. L.J. 753, 753 (2013) (arguing that open innovation movements' reliance on copyright to promote their movements leads to unintended anticommons problems).

[23] *See* Heller, *supra* note 17, at 625.

[24] *See, e.g.*, F. Scott Kieff, *IP Transactions: On the Theory & Practice of Commercializing Innovation*, 42 HOUS. L. REV. 727, 740–42 (2005) (questioning the logic of the anticommons narrative as applied to software and other industries); Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961, 999–1009 (2005) (arguing that patent rights do not lead to an anticommons in the software industry); Mark Schankerman & Michael Noel, *Strategic Patenting and Software Innovation*, 61 J. INDUS. ECON. 481, 514 (2013) (discussing the role of patent thickets in deterring innovation in the software industry); James Bessen & Eric Maskin, *Sequential Innovation, Patents, and Imitation* 2 (M.I.T. Dep't of Econ., Working Paper No. 00-01, 2000) (arguing that strong and varied patent rights in the software industry hinder sequential and complementary innovation); James Bessen, *Patent Thickets: Strategic Patenting of Complex Technologies* (Bos. Univ. Sch. of Law, Working Paper, 2003), http://ssrn.com/abstract=327760 (arguing that patent thickets discourage innovation).

[25] *See, e.g.*, *supra* note 24.

[26] *See* David Fagundes & Jonathan S. Masur, *Costly Intellectual Property*, 65 VAND. L. REV. 677, 718 (2012) (indicating that "for the most part, microworks do not present a significant risk of welfare-diminishing holdouts. This is because the numerous limitations on owners' exclusive rights and opportunities for users to work around those rights"); Michael A. Heller, *The Boundaries of Private Property*, 108 YALE L.J. 1163, 1175 n.61 (1999) (arguing that copyright is less prone to cause anticommons concerns because of copyright law's fair use doctrine).

[27] *See, e.g.*, Chase A. Marshall, *A Comparative Analysis: Current Solutions to the Anticommons Threat*, 12 J. HIGH TECH. L. 487, 503–04 (2012) (describing the open source software licensing model as a potential cure to anticommons problems in the software industry); Mike Loukides, *Avoiding the Tragedy of the Anticommons*, O'REILLY RADAR (Oct. 23, 2014), http://radar.oreilly.com/2014/10/avoiding-the-tragedy-of-the-anticommons.html (arguing that the sciences can avoid an anticommons by adopting a model of innovation similar to the open source software licensing movement).

[28] *See, e.g.*, Anthony L. Clapes, Patrick Lynch & Mark R. Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493, 1501 (1987) (assessing the proper scope of copyright as applied to software); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1050 (1989) (arguing that

This Article argues that one reason such line-drawing questions are important is because a line that yields more copyrights will result in greater anticommons problems in today's collaborative software world.[29] In contrast to previous scholarship, this Article thus argues that the collaborative nature of modern software innovation may result in significant anticommons concerns because that collaboration increases the number of copyrights in any given software product. And decisions such as *Oracle v. Google*, which may expand the reach of software copyright, will likely exacerbate these types of anticommons problems.

A second, related trend that is likely to lead to growing anticommons concerns in the software world is the interconnected nature of much modern software innovation. Indeed, in today's world, a growing need exists for software products to interoperate with each other, particularly as more and more software moves into the "cloud" in order to facilitate the "Internet of Things" economy. For instance, the growing Internet of Things requires that more and more software products be hosted on the computers of companies (i.e., in the cloud), rather than on those of users, to provide Internet access to those services on a variety of interconnected products. Yet in order for these heterogeneous products and services to successfully interoperate with each other in exchanging information, they must also typically exchange software interfaces. And in order for those exchanges to occur, parties must be willing to make these software interfaces available for use by third parties. While in many cases parties have incentives to do exactly that, in other cases they may not. Indeed, Google incorporated the Java APIs into Android based in part on a desire to facilitate interoperability in an interconnected world.[30] And it did so with the acquiescence of the Java APIs' owner at the time, Sun

---

the copyright idea/expression merger doctrine should allow unlicensed use of standardized interfaces and diffusion of scientific ideas); A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection of Computer Programs*, 72 NEB. L. REV. 351, 358 (1993) (arguing that patents are more appropriate for software protection than is copyright); Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2312–13 (1994) [hereinafter *Manifesto*] (proposing a sui generis regime for software because of its unique characteristics).

[29] *Cf.* Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 346–47 (1970) (arguing that extension of copyright protection to all computer programs may result in significant "transaction cost" problems because many computer users copy small portions of programs for reuse and modification, and needing to obtain permission for such uses in each case may become prohibitively expensive).

[30] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *aff'd in part, rev'd in part*, 750 F.3d 1339 (Fed. Cir. 2014).

Microsystems.[31] It was only later that the subsequent owner of the APIs, Oracle, objected to Google's use of the APIs.[32]

Hence, anticommons concerns increase in the interconnected world in part because interoperability needs mean any given device or service will seek to incorporate software from other parties in order to facilitate that interoperability. These interconnected products and services, therefore, may also become subject to the copyright interests of hundreds and thousands. And when they do, similar holdup and transaction cost concerns result as described above.

This Article thus argues that these two interrelated trends mean that copyright and the theories behind it are increasingly ill-equipped to foster software innovation in today's world. The majoritarian theory behind modern copyright law, for instance, fails to fully account for the collaborative, interconnected realities of much modern software development.[33] Indeed, today's software development norms conflict with the norms and operation of copyright law, which emphasize the interests of copyright holders[34] while traditionally paying insufficient attention to the dynamics of incremental software development among a variety of stakeholders.[35] Copyright law's majoritarian theory, which posits that software creators will not incur the costs necessary to develop software without exclusive rights in that software,[36] is thus increasingly unpersuasive in the cloud and Internet of Things economies,[37] which rely on shared resources[38] and interoperability between heterogeneous computing devices and services.[39]

---

[31] Dan Farber, *Former Sun CEO Says Google's Android Didn't Need License for Java APIs*, CNET (Apr. 26, 2012, 11:38 AM), http://www.cnet.com/news/former-sun-ceo-says-googles-android-didnt-need-license-for-java-apis/ (discussing Java adoption generally as one of the reasons that Sun, the previous owner, may have given Google a free pass on using the company's APIs despite the lack of a license).

[32] *Id.*

[33] *See* Shyamkrishna Balganesh, *Foreseeability and Copyright Incentives*, 122 HARV. L. REV. 1569, 1576–77 (2009) (describing the utilitarian majoritarian theory behind copyright law as providing creators with the proper incentives to create original works of authorship); *accord* William M. Landes & Richard A. Posner, *An Economic Analysis of Copyright Law*, 18 J. LEGAL STUD. 325, 326 (1989).

[34] Balganesh, *supra* note 33; Landes & Posner, *supra* note 33.

[35] Asay, *supra* note 22; *see also* Clark D. Asay, *Enabling Patentless Innovation*, 74 MD. L. REV. 431 (2015); Liza S. Vertinsky, *Making Room for Cooperative Innovation*, 41 FLA. ST. U. L. REV. 1067 (2014) (making a similar argument with respect to patent law's failure to take into account the dynamics of cooperative innovation).

[36] Balganesh, *supra* note 33, at 1576–77.

[37] Antonio Regalado, *The Economics of the Internet of Things*, MIT TECH. REV. (May 20, 2014), http://www.technologyreview.com/news/527361/the-economics-of-the-internet-of-things/ (arguing that the Internet of Things will largely be fueled by the proliferation of platforms).

What this Article does not suggest is that software innovation will grind to a halt without radically changing copyright law as applied to software; parties have plenty of other reasons to continue to pursue software innovation, and those reasons will likely continue to result in software innovation.[40] Indeed, copyright law's traditional premises may still apply in many cases in explaining the motivations of parties in pursuing software innovation.[41] But in many other cases the robustness of software innovation, and information technology innovation more generally, is likely to decrease as parties seek to navigate copyright "thickets" in pursuing collaborative software innovation in an interconnected world.

This Article thus argues for better adapting copyright law to software innovation's modern dynamics. While several possibilities exist for making these adaptations, this Article assesses the merits of copyright's fair use defense for this role. The Article focuses on fair use for several reasons. First, fair use is copyright law's most important defense to copyright infringement.[42] Fair use is thus at the center of many copyright controversies, and it is hoped this Article's analysis proves useful in assessing fair use in current and future copyright controversies involving software. Second, the flexible, broad nature of the fair use inquiry lends itself to adaptation.[43] For instance, fair use has taken on new, important roles over time.[44] And third, adapting fair use to better take into account modern software innovation's collaborative, interconnected realities arguably serves the primary purpose of the fair use doctrine, which is

---

[38] Asay, *supra* note 22 (describing how copyright law may work to inhibit the use and development of FOSS resources based on fears of copyright infringement).

[39] Reuven Cohen, *Cloud Interoperability and the Battle for the Open Cloud*, FORBES (Apr. 26, 2013, 3:38 PM), http://www.forbes.com/sites/reuvencohen/2013/04/26/cloud-interoperability-and-the-battle-for-the-open-cloud/2/ (discussing the need for greater openness and standardization in cloud technologies in order to enable them to achieve greater interoperability and thereby achieve the cloud's potential).

[40] *See, e.g.*, Stuart J.H. Graham et al., *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L.J. 1255, 1290 fig.1 (2009) (highlighting the importance of "first-mover" advantages to many software innovators).

[41] *See, e.g.*, Pamela Samuelson, *The Uneasy Case for Software Copyrights Revisited*, 79 GEO. WASH. L. REV. 1746 (2011) (describing the role copyright appears to have played and may continue to play in encouraging software innovation, while also noting eight modern trends that may make software copyright less relevant today).

[42] Andrew Inesi, *A Theory of De Minimis and a Proposal for Its Application in Copyright*, 21 BERKELEY TECH. L.J. 945, 983–84 (2006) (discussing case law articulating the purposes behind the fair use doctrine).

[43] Michael W. Carroll, *Fixing Fair Use*, 85 N.C. L. REV. 1087, 1149 (2007) (recognizing that too certain of rules in the fair use inquiry would hamper one of the doctrine's virtues, its flexibility); Ned Snow, *Judges Playing Jury: Constitutional Conflicts in Deciding Fair Use on Summary Judgment*, 44 U.C. DAVIS L. REV. 483, 497–98 (2010) (indicating that fair use is a "flexible doctrine").

[44] Rebecca Tushnet, *Content, Purpose, or Both?*, 90 WASH. L. REV. 869, 869–71 (2015).

to allow for uses that promise to boost, rather than diminish, creativity overall.[45] Other potential solutions, such as a compulsory licensing scheme, are also briefly considered.

This Article has four parts. Part I reviews the traditional account of copyright's role in fostering software innovation. Part II then challenges this traditional account in light of the considerations briefly discussed above, namely, the increasingly collaborative, interconnected nature of software development, particularly as the cloud and Internet of Things economies grow. It suggests these trends mean that copyright as applied to software is likely to result in growing anticommons problems, and it examines the *Oracle v. Google* case as an example of these problems in practice. Part III then proposes and assesses changes to copyright law's fair use defense in light of these developments and argues that such changes could help better promote of software innovation in the future. Part IV briefly postulates that similar adaptations may be warranted in other fields of creativity where collaborative, interconnected development practices are increasingly common. The Article then concludes.

I. WHY SOFTWARE COPYRIGHT?

This Part lays out the economic basics of how copyright law is meant to encourage software innovation, and how software came to be subject to copyright at all. It also briefly explores some of the main concerns that others have raised over the years about copyright as applied to software.

A. The Early Days

Others have chronicled the history of how software came to be subject to copyright protection.[46] This section does not cover that history in full, but instead provides a snapshot of it in order to set the stage for the rest of this Article's analysis.

At first glance, it may seem strange that software is copyrightable subject matter at all. Copyright, after all, is meant to protect and encourage expressive, creative activities.[47] Expressive, creative activities are perhaps more readily

---

[45] Inesi, *supra* note 42, at 983.

[46] *See, e.g.*, Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663 [hereinafter *CONTU Revisited*].

[47] *See generally* 17 U.S.C. § 106 (2012) (setting forth the basic rights of copyright holders in their original expression of ideas).

seen in domains such as literature, music, and movies. And these types of works differ from software in important respects, the most important of which is that software is essentially functional in nature, while these other types of works are not. For instance, software's general purpose is to enable a computing device to perform some function as efficiently as possible, even if the output of the software is some creative object.[48] Software's primary aim, therefore, is to provide a utilitarian solution to some sort of computing problem. Scholars have traditionally viewed patent law, as opposed to copyright law, as the appropriate body of law for encouraging and protecting these types of utilitarian solutions.[49]

Yet software does include expressive elements that in some ways are similar to those found in literary, musical, and other creative works. For instance, software engineers initially write most software programs in what is called "source code."[50] Source code is the human-readable version of software programs.[51] At least, it is readable by those familiar with the relevant programming language in which the source code was written. This source code is then translated by a compiler into computer-readable "object code."[52] Once this translation occurs, the computing device is able to understand the code's instructions and perform whatever functions the code dictates.

Hence, source code is clearly expressive in the sense that computer engineers utilize various computer programming languages to express a variety of functional computer instructions. And the object code, which is simply a translation of that source code into a version that a computer can execute, contains essentially the same expression, just in a machine-readable form.

These and related considerations seem to have led the U.S. Copyright Office to register some copyrights in software early on, before it was either judicially or legislatively certain that software was subject to copyright.[53] But

---

[48] Robert Plotkin, *Computer Programming and the Automation of Invention: A Case for Software Patent Reform*, 7 UCLA J.L. & TECH., Fall 2003, at 1, 5–7 (providing an extensive definition and discussion of what constitutes "software" and the purposes behind it).

[49] Clark D. Asay, *Intellectual Property Law Hybridization*, 87 U. COLO. L. REV. 65, 68 (2016).

[50] The Linux Information Project, *Source Code Definition*, LINFO, http://www.linfo.org/source_code.html (last updated Feb. 14, 2006).

[51] *Id.*

[52] *Id.*

[53] Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEX. L. REV. 1921, 1947–48 (2007) (detailing how the Copyright Office registered some copyrights in software before the 1976 Copyright Act despite doubting that computer programs were subject to copyright protection at all).

passage of the Computer Software Copyright Act of 1980 dispelled any uncertainty regarding whether copyright applied to software. This Act amended the Copyright Act of 1976 to expressly include a definition of "computer program" under the Copyright Act, in accordance with recommendations from the National Commission on New Technological Uses of Copyrighted Works (CONTU) Commission.[54] The CONTU Commission had been formed in the run-up to the 1976 Copyright Act to study what changes to copyright law were necessary in light of advancements in computer technologies.[55] Its recommendations, which were made after the Copyright Act of 1976 had already been enacted, came down strongly in favor of software programs being subject to copyright,[56] though the Commission left it up to courts to work out the exact scope of that protection.[57]

In arguing in favor of copyright protection for software, the Commission reasoned that "if the cost of duplicating information is small, then it is simple for a less than scrupulous person to duplicate it."[58] According to the Commission, these economic realities meant that "legal as well as physical protection" for software programs were necessary to incentivize parties to create and disseminate them.[59] The Commission backed up this economic reasoning by pointing to two trends in the computer industry at the time. First, parties were increasingly writing software programs to perform multiple functions on a variety of computers, rather than writing more rudimentary programs that performed a limited number of functions on only a specific computer model.[60] In other words, an independent mass-market for software products had developed.

Second and importantly in the Commission's view, those writing these software programs were increasingly different parties than those building the actual computers running the software.[61] When the same party built and sold the hardware and software together, that party was able to recoup its software development costs through hardware sales.[62] But independent software

---

[54] *CONTU Revisited*, *supra* note 46, at 694.

[55] *Id.* at 694–95.

[56] See NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 9–10 (1978).

[57] *Id.* at 22–23.

[58] *Id.* at 10–11.

[59] *Id.*

[60] *Id.*

[61] *Id.* at 11.

[62] *Id.*

developers must recoup their costs, if at all, through sale of the software.[63] As such, the growing software industry was unlikely to survive absent copyright protection because "[t]he cost of developing computer programs is far greater than the cost of their duplication."[64] To encourage the creation and dissemination of software programs, therefore, the Commission recommended that copyright apply to software.[65]

According to some accounts, copyright protection for software has helped spur significant innovation in the software industry. For instance, one study notes that copyright played a "nontrivial" role in spurring innovation in the software industry between 1970 and 2000, when the industry experienced phenomenal growth.[66] Copyright helped encourage software innovation during this time period largely because of the reasons the CONTU Commission highlighted: software programs had increasingly become divorced from specific computer models or users and instead were mass-marketed to heterogeneous parties for use in a variety of computing environments.[67] Copyright thus helped this market flourish by providing developers with exploitable economic rights in their software products.[68]

B. Dissenting Views

Despite this optimistic view of copyright's role in facilitating software innovation, scholars have long worried that copyright's application to software remains an odd fit.[69] The utilitarian nature of software is largely to blame for these concerns, because the functional nature of software makes identifying the proper scope of software copyright difficult. But properly delineating that boundary is crucial to ensuring robust software innovation.[70]

---

[63] *Id.*

[64] *Id.*

[65] *Id.* at 11–12.

[66] Samuelson, *supra* note 41, at 1757–65.

[67] *Id.*

[68] *Id.*

[69] *See, e.g.*, Stacey L. Dogan & Joseph P. Liu, *Copyright Law and Subject Matter Specificity: The Case of Computer Software*, 61 N.Y.U. ANN. SURV. AM. L. 203 (2005) (demonstrating that despite the general trend among courts of significantly adapting copyright doctrines to deal with the special features of computer software, a few courts and legislatures have adopted a more rigid approach, and contending that increased flexibility and adaptation is critical in the software context).

[70] *See, e.g.*, Jacqueline D. Lipton, *IP's Problem Child: Shifting the Paradigms for Software Protection*, 58 HASTINGS L.J. 205, 206, 240 (2006) ("[C]opyrights can serve to chill innovation unless clearer guidelines about copyright limitations in the software context are developed.").

For instance, Pamela Samuelson has argued that the machine-readable form of software does not merit copyright protection because, among other reasons, machine-readable software is inherently utilitarian.[71] Unlike instructions in a book that inform the reader how to perform a task, the machine-readable version of software actually performs the task.[72] This type of intrinsic utility thus differs from the types of utility that copyright is meant to cover, such as conveying information or portraying an appearance.[73] The functional nature of software, therefore, demands specific copyright limitations when applied to software—and perhaps a different legal regime altogether—in order to best foster software innovation.[74]

Other scholars have proposed additional software copyright boundaries with a view to better fostering software innovation. For instance, Peter Menell has argued that copyright law's idea/expression dichotomy should be liberally applied to software in order to eliminate copyright protection for software interfaces and scientific ideas.[75] The basic purpose behind copyright law's idea/expression doctrine is to limit copyright protection to the expression of ideas, while excluding the underlying ideas from the scope of copyright.[76] For instance, copyright may apply to an author's particular articulation of a system of accounting as expressed in a book, but not to the ideas underlying the accounting system.[77] Hence, while copyright prohibits third parties from copying the book, it does not prohibit those same third parties from practicing the ideas behind the accounting system.[78]

According to Menell, applying this idea/expression dichotomy liberally so that copyright does not extend to software interfaces and scientific principles underlying software would better promote software innovation by ensuring the development and diffusion of these interfaces and ideas.[79] Dennis Karjala has

---

71 *CONTU Revisited*, *supra* note 46, at 727–28.

72 *Id.* at 727.

73 *Id.* 727–28.

74 *See id.*

75 Menell, *supra* note 28, at 1047–50.

76 Ashton-Tate Corp. v. Ross, 728 F. Supp. 597, 601 (N.D. Cal. 1989), *aff'd*, 916 F.2d 516 (9th Cir. 1990) ("The foundation of federal copyright law is that only expressions of ideas, not the ideas themselves, give rise to protected interests.").

77 Baker v. Selden, 101 U.S. 99. 104–05 (1879).

78 *Id.* at 104, 107.

79 Menell, *supra* note 28, at 1047–50; *see also* Pamela Samuelson, *Guest Post: Are APIs Patent or Copyright Subject Matter?*, PATENTLY-O (May 12, 2014), http://patentlyo.com/patent/2014/05/copyright-subject-matter.html (reviewing some of the leading cases that seek to address these problems); Pamela Samuelson, *The Strange Odyssey of Software Interfaces and Intellectual Property Law* (Berkeley Ctr. for Law

made similar suggestions in advocating scaling back copyright protection for software in order to better promote software innovation.[80] Others have pointed to copyright's fair use defense as a largely untapped means of pushing back against expansive software copyright by enabling greater reuse of otherwise copyrighted software.[81]

Some studies even go so far as to suggest that the proper boundary between copyrightable expression and non-copyrightable ideas is impossible to consistently identify in the software context. For instance, one prominent proposal advocates a sui generis legal regime for software because of concerns that copyright as applied to software leads to cycles of under- and over-protection.[82] Such cycles, according to these scholars, are largely the result of judges struggling and ultimately failing to apply traditional copyright doctrines to the functional world of software.[83] And this failure is in some sense inevitable because copyright simply cannot be adequately tailored to software without morphing copyright law into something else entirely.[84] A different legal regime is thus necessary given the distinct characteristics of software.[85] Others have come to similar conclusions.[86]

Even more drastically, some within the software industry have argued against applying copyright to software at all. For instance, some in the "Free Software" and "Open Source Software" movements have expressed this sentiment, though, ironically, these movements rely on copyright in pursuit of their aims.[87] In general, these movements are founded on the belief that copyright grants to software copyright holders excessive control that hinders,

---

& Tech, Paper No. 59, 2008), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1323818 (tracing the evolution of intellectual property law protection for software application programming interfaces).

[80] Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33, 94–96 (1987) (arguing against what he views as courts' expansive interpretation of copyright as applied to software and advocating curtailing software copyright based on a policy of anti-piracy).

[81] Pamela Samuelson, *Fair Use for Computer Programs and Other Copyrightable Works in Digital Form: The Implications of* Sony, Galoob *and* Sega, 1 J. INTELL. PROP. L. 49, 84–86 (1993).

[82] *Manifesto*, *supra* note 28, at 2312, 2356–61.

[83] *Id.* at 2356–61.

[84] *Id.*

[85] *Id.* at 2357.

[86] *See, e.g.*, Dennis S. Karjala, *Protecting Innovation in Computer Software, Biotechnology, and Nanotechnology*, 16 VA. J.L. & TECH. 42, 47 (2011) (arguing that applying copyright directly to solve the software misappropriation problem was a mistake, and that "[i]n retrospect, a sui generis statute that protected only literal or near-literal copying of code, for a much shorter term, would have been preferable").

[87] *See* Chris DiBona et al., *Introduction* to OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION (Chris DiBona, Sam Ockman & Mark Stone eds., 1999) (providing a comprehensive history of the beginnings of these movements).

rather than promotes, software innovation.[88] Most famously, Richard Stallman, the founder of the Free Software movement, has argued that software should not be subject to copyright ownership because software consists of utilitarian ideas, and controlling ideas constitutes control over the lives of others.[89] In order to counteract the negative consequences of software copyright, leaders of these movements have devised copyright licenses that are meant to free software from the typical controls that copyright confers.[90]

Hence, though some may believe that the CONTU Commission largely got it right,[91] others, as discussed above, have articulated lingering concerns over copyright being applied to software in ways that frustrate software innovation. And these concerns typically focus on determining the proper boundary between copyrightable expression and non-copyrightable ideas, which the functional nature of software makes inherently difficult.

II. COPYRIGHT AND THE MODERN SOFTWARE INDUSTRY

The previous Part laid out the typical economic rationales in favor of applying copyright protection to software, as well as some of the more prominent critiques of doing so. Those critiques generally argue that the utilitarian nature of software makes applying traditional copyright law principles to software difficult, and they suggest a variety of legal reforms aimed at better tailoring the law so that it takes into account software's unique functional characteristics. In some cases, these proposals suggest a new legal regime for software altogether.

This Part argues that at least two growing trends in modern software development practices present a different, though related, copyright challenge for software innovation. First, more and more software is collaboratively built, which means that any given piece of software may include dozens, hundreds, or even thousands of copyright holders. Second, in today's cloud and Internet of Things economies, a growing need exists for heterogeneous products and services to exchange software interfaces in order to interoperate and share data.

---

88 *See, e.g.*, RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 40–41, 45–56 (2d ed. 2010) (arguing that software "freedom" enables developers to share their improvements with each other more readily, which in turn leads to enhanced innovation).

89 *Id.* at 33–34.

90 Asay, *supra* note 22, at 759–62.

91 Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 982–83 (1993) (defending CONTU recommendation that software should be copyrightable).

Satisfying this need thus also means that any given software product or service is more likely to include copyrightable software from third parties because use of third-party software interfaces is necessary to enable the desired interoperability.

These trends together mean that anticommons issues are more likely to arise simply because a prospective user of a collaboratively-built, interconnected software resource must navigate more copyright interests before she can use the resource. Though the liberal nature of the licensing regimes that govern much collaboratively built software may help reduce these concerns in many cases, they do not effectively eliminate them. Indeed, these licensing regimes can themselves also result in underutilization of the software resources, as described more fully below.

These anticommons concerns are thus crucially related to concerns over the proper scope of software copyright as previously articulated by others. These concerns are related because expanding the scope of software copyright also increases anticommons problems by increasing the number of copyrights in any given piece of software. Indeed, expanding the scope of software copyright compounds possible anticommons concerns that the collaborative, interconnected nature of modern software innovation leads to. The following sections review these trends and then show how they create anticommons concerns. And as will be discussed, collaborative licensing schemes do not eliminate such concerns, but instead in some instances actually help create them.

A. Software's Collaborative Milieu

Over the years, software innovation has undergone a radical transformation. As discussed above, early software development practices provided less justification for subjecting software to copyright because software developers generally had means by which to recoup their development costs, such as through fees charged for customizing the software or accompanying hardware sales.[92] But as a mass-market for multi-functional, independent software programs developed, software copyright arguably became justified as a necessary means by which software vendors recouped their costs of developing the software programs.[93]

---

[92] Samuelson, *supra* note 41, at 1757–65.

[93] *Id.*

The software industry has once again changed, and dramatically so. One key change was the development and widespread adoption of object-oriented programming.[94] To oversimplify, object-oriented programming is a "building-block" approach to software development.[95] In this approach, software is written as a series of "objects." These objects are self-contained, meaning they can function independently, and they are "modular," meaning that other software objects can be created and used with them without having to completely rewrite the preexisting software objects.[96] Hence, if a software developer wants to write a new software program using object-oriented programming, she can simply select a group of preexisting software objects and combine them in a new way, add some new software objects for interacting with them, or create a new set of software objects altogether.[97]

For purposes of this Article, several important implications arise from the widespread adoption of object-oriented programming in the software world, which adoption began in earnest in the 1980s.[98] First, object-oriented programming made collaborative software innovation more likely because object-oriented programming's modularity and self-contained nature meant that a variety of parties could create software objects capable of working together, so long as the relevant software interfaces—which enable the different software objects to interact—were made available.[99] Second and related, object-oriented programming made collaborative software innovation more desirable because the modularity and self-contained nature of object-oriented programs meant that collaboration could dramatically increase the pace of software innovation.[100] For instance, software programmers need not completely reprogram their own programs in order to build on the work of

---

[94] *See generally* BERTRAND MEYER, OBJECT-ORIENTED SOFTWARE CONSTRUCTION (2d ed. 1997) (describing the development and adoption of object-oriented programming).

[95] Michael A. Dryja, *Looking to the Changing Nature of Software for Clues to its Protection*, 3 U. BALT. INTELL. PROP. L.J. 109, 126 (1995).

[96] Joshua A.T. Fairfield, *BitProperty*, 88 S. CAL. L. REV. 805, 852–53 (2015) (describing these characteristics of object-oriented programming); Henry E. Smith, *Institutions and Indirectness in Intellectual Property*, 157 U. PA. L. REV. 2083, 2088, n.16 (2009) (describing the virtues of modularity).

[97] Dryja, *supra* note 95.

[98] Plotkin, *supra* note 48, at 34.

[99] *See generally* Keith Stephens & John P. Sumner, *Software Objects: A New Trend in Programming and Software Patents*, 12 SANTA CLARA COMPUTER & HIGH TECH. L.J. 1, 4–12 (1996) (discussing some of these benefits of object-oriented programming).

[100] Lipton, *supra* note 70, at 227–28 (discussing copyright as a challenge to this mode of programming given its focus on reusing software objects).

others, instead relying on interfaces to enable their works and those of others to interoperate.[101]

Not coincidentally, the free and open source software (FOSS) movement also began in earnest in the 1980s.[102] The movement started as a reaction to copyright holders exerting their rights in ways that prevented developers from improving upon copyrighted software.[103] Figures such as Richard Stallman, the so-called "prophet" of the Free Software movement, responded by devising copyright licensing schemes that inverted copyright.[104] For instance, leaders of the FOSS movement developed and released software to the public under copyright licenses that enabled third parties to use the software in generally liberal ways, so long as those parties complied with the software's license conditions.[105]

The most famous of these conditions, often called "copyleft," means that subsequent users and distributors of the software must release any changes they make to the software subject to the same terms that initially governed the software.[106] One of the primary purposes of copyleft licensing was thus to ensure that FOSS remained subject to FOSS licensing terms, as well as requiring users of the software resources to contribute software back to the community under the same terms.[107] These licensing mechanics therefore helped spread the FOSS movement's norms of collaboration and software freedom, which were in some ways a natural fit within a software industry increasingly devoted to object-oriented programming and its building-block approach to software development.

Of course, it should be made clear that the FOSS movement is by no means a unitary one. Debates between different factions of the movement have raged for some time as to what the movement's goals should be and how best to

---

101 *Id.*

102 *See generally* ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR (rev. ed. 2001); Simon Phipps, *The Rise and Rise of Open Source*, INFOWORLD (May 8, 2015), http://www.infoworld.com/article/2914643/open-source-software/rise-and-rise-of-open-source.html ("The results from the annual Future Of Open Source survey are in, and they confirm everything we already knew: Open source is now the default.").

103 *See generally* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 87 (providing a comprehensive history of the beginnings of the free and open source software movement).

104 *Id.* at 2–3.

105 Asay, *supra* note 22, at 759–62.

106 *Id.* at 760.

107 Eben Moglen, *Anarchism Triumphant: Free Software and the Death of Copyright*, 4 FIRST MONDAY, Aug. 2, 1999, at 1, 20–22, http://firstmonday.org/ojs/index.php/fm/article/view/684/594.

achieve those goals.[108] But importantly for purposes of this Article, at least one common theme underlies the movement, regardless of faction: a collaborative innovation model is superior to the siloed approaches that had frustrated so many early FOSS developers. A famous refrain from one of the FOSS movement's leaders makes the purported supremacy of this approach clear: "[g]iven enough eyeballs, all [software] bugs are shallow."[109] The licensing schemes that early leaders of the FOSS movement spearheaded were thus meant to champion collaborative software innovation over a go-it-alone approach. And importantly, that collaborative approach to software innovation aligned well with the growing adoption of object-oriented software programming more generally.

According to many accounts, this collaborative, object-oriented approach to software innovation has largely prevailed.[110] Indeed, though early on entrenched players in the software industry were skeptical and sometimes even hostile to the FOSS movement,[111] the software industry has, by and large, come to embrace a collaborative innovation model,[112] with some now even arguing that it is the software industry's default innovation paradigm.[113] Utilizing this paradigm, parties across the globe have collaboratively built some of the most popular and important software technologies in the world, including Linux, Android, Apache Web Server, Firefox, Netflix, Airbnb, and many others that power much of the Internet and computing world.[114] Even

---

[108] *See, e.g.*, Clark D. Asay, *The General Public License Version 3.0: Making or Breaking the FOSS Movement?*, 14 MICH. TELECOMM. & TECH. L. REV. 265, 267–71 (2008) (reviewing the differences between some of the more important factions within the broader FOSS movement, while concluding that the factions have more in common than often meets the eye).

[109] ERIC STEVEN RAYMOND, *Release Early, Release Often*, *in* THE CATHEDRAL AND THE BAZAAR 38, 41 (2000), http://www.catb.org/~esr/ writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html.

[110] Baldwin & von Hippel, *supra* note 19 (arguing that collaborative models of open innovation are increasingly displacing "producer" models of innovation, in software and elsewhere); Phipps, *supra* note 102; Snow et al., *supra* note 19, at 8 (noting that the study of collaborative models of innovation has focused primarily on open source software communities, while arguing that these innovation models are increasingly applicable elsewhere as well).

[111] Thomas C. Greene, *Ballmer: "Linux Is a Cancer"*, THE REGISTER (June 2, 2001, 6:19 PM), http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/ (referencing Steve Ballmer, then-CEO of Microsoft, and his now infamous remark that FOSS is akin to cancer); Graham Lea, *MS' Ballmer: Linux Is Communism*, THE REGISTER (Jul. 31, 2000, 10:10 PM), http://www.theregister.co.uk/2000/07/31/ms_ballmer_linux_is_communism/ (referring to Ballmer and his likening Linux, the most successful FOSS project at the time, to a form of communism).

[112] Baldwin & von Hippel, *supra* note 19, at 1411–12; Snow et al., *supra* note 19, at 8.

[113] Phipps, *supra* note 102.

[114] Doug Dineley, *The Greatest Open Source Software of All Time*, INFOWORLD (Aug. 17, 2009), http://www.infoworld.com/article/2631146/open-source-software/the-greatest-open-source-software-of-all-time.html

Microsoft, long perceived as the arch nemesis of the FOSS movement has come to embrace certain collaborative development practices.[115] In fact, FOSS software options are increasingly displacing many previously dominant proprietary software offerings.[116] And many experts project that this trend will only continue.[117]

These developments have dramatically altered the realities of copyright as applied to software. For instance, before collaborative models of innovation were widely used, a software product typically had one author: the party that developed the software product. This was so even in corporate settings where many employees of a corporation helped create the software product because corporate employees typically, as part of their employment contracts, assign any rights they have in the software to their employer.[118] Indeed, an important function of copyright's "work-made-for-hire" doctrine is precisely to make it simpler for employers to obtain the copyrights in the collective contributions of their employees and independent contractors.[119] The typical result was a fairly clean copyright story: a company, despite employing hundreds or even thousands of software engineers, was the sole author of the copyrighted software. Thus, to obtain access to that software, a third party need only transact with that single company.

(listing some of the most famous FOSS technologies of all time, including Linux, and their important role in powering much of the computing world).

115 *See, e.g.*, Steven J. Vaughan-Nichols, *Microsoft: The Open-Source Company*, ZDNET (Jan. 26, 2015, 7:04 PM), http://www.zdnet.com/article/microsoft-the-open-source-company/ (describing Microsoft's relative embrace of free and open source software).

116 *See, e.g.*, Kurt Marko, *Red Hat's CEO Sees Open Source Cloud Domination*, FORBES (June 8, 2014, 8:16 PM), http://www.forbes.com/sites/kurtmarko/2014/06/08/red-hat-ceo-open-source-clouds/ (noting that "outside of Microsoft Azure, the underlying infrastructure of all the major public cloud services is built upon open source software" in arguing that FOSS software will increasingly displace proprietary solutions in so-called "Cloud Computing"); Katherine Noyes, *How a Little Open Source Project Came to Dominate Big Data*, FORTUNE (June 30, 2014, 5:49 PM), http://fortune.com/2014/06/30/hadoop-how-open-source-project-dominate-big-data/ (describing the rise of one such FOSS project, Hadoop, and how it has come to be the industry standard for so-called "Big Data").

117 Marko, *supra* note 116; Noyes, *supra* note 116; *see also* Matthew N. Asay, *Open Source and the Commodity Urge: Disruptive Models for a Disruptive Development Process*, *in* OPEN SOURCES 2.0: THE CONTINUING EVOLUTION 103, 103–04 (Chris DiBona, Danese Cooper & Mark Stone eds., 2006) (noting that FOSS solutions will continue to "commodify" technologies all along the software stack, while also indicating that proprietary vendors will still be able to reap profits at the top of the software chain where solutions will remain non-commodities).

118 *See generally* ORLY LOBEL, TALENT WANTS TO BE FREE (2013) (detailing the many ways in which employers obtain rights to the intellectual contributions of their employees, including by way of typical invention assignment agreements).

119 *See* Matthew Vincent H. Noller, Note, *Darkness on the Edge of Town: How Entitlements Theory Can Shine a Light on Termination of Transfers in Sound Recordings*, 46 GA. L. REV. 763, 784 (2012).

Today's collaborative software world presents a much more complicated copyright picture. For instance, rather than a software product having a single author, software products today may include hundreds and even thousands of copyright holders, each with a separate copyright interest. Object-oriented programming makes this particularly likely since programs often depend on and include a variety of software objects from third parties.[120] To illustrate: the Linux kernel, which provides the backbone for much modern-day computing, includes thousands of distinct contributors, and each of those contributors may possess a copyright interest in some portion of their contribution.[121] Other popular, collaboratively built software products include numerous copyright holders as well.[122] What is more, each of these projects may be and often is combined with other software objects that similarly include dozens, hundreds, or thousands of additional copyright interests.[123] This reality also applies in situations where a company produces and distributes proprietary software products because even these products increasingly include collaboratively built software.[124]

The result is that parties wishing to make use of modern software technologies typically face a more complicated copyright situation than parties in earlier eras. Rather than dealing with a single copyright holder, software's modern collaborative milieu means that using today's software products typically implicates multiple copyright interests. And the multiplicity of these interests can lead to significant anticommons concerns, as described more fully below in section C.

---

[120] James Y. Song, *Searching for a Link Between Software Patent and Object-Oriented Programming*, 76 J. PAT. & TRADEMARK OFF. SOC'Y 687, 687–90 (1994) (discussing the concepts of "dependence" and "inheritance" within object-oriented programming).

[121] *Torvalds/Linux*, *supra* note 20 (listing over 5500 distinct contributors to the project).

[122] *See, e.g.*, *Apache/Hadoop*, GITHUB, https://github.com/apache/hadoop (last visited Oct. 21, 2015) (listing seventy-nine distinct contributors to the project); *Mongodb/Mongo*, GITHUB, https://github.com/mongodb/mongo (last visited Oct. 21, 2015) (listing 264 distinct contributors to the project); *PHP/PHP-src*, GITHUB, https://github.com/php/php-src (last visited Oct. 21, 2015) (listing 381 distinct contributors to the project).

[123] *See, e.g.*, Jerry Hildenbrand, *What Is Android?*, ANDROIDCENTRAL (May 16, 2015, 4:36 PM), http://www.androidcentral.com/what-android (laying out the basic architecture of Google's Android, which includes numerous different FOSS projects that are ultimately combined into one to provide for Android's functionalities).

[124] *See, e.g.*, Ben Kepes, *Open Source Is Good and All, but Proprietary Is Still Winning*, FORBES (Oct. 2, 2013, 2:41 PM), http://www.forbes.com/sites/benkepes/2013/10/02/open-source-is-good-and-all-but-proprietary-is-still-winning/ (describing instances where Oracle has combined FOSS software with its own proprietary technology to produce a commercial product).

Of course, copyright includes a number of doctrines meant to reduce the number of potential claimants. The work-made-for-hire doctrine, as mentioned above, is one such concept. Others may include the joint work doctrine, which attempts to limit who can be an author of a work that includes more than one contributor.[125] Indeed, as the Ninth Circuit ultimately found in the *Garcia* decision mentioned in the Introduction, not everyone that contributes some creative expression to a larger work can claim a copyright interest in that contribution.[126]

While these doctrines may help in some cases, they certainly do not make the copyright messiness disappear. The work-made-for-hire doctrine, for instance, is simply inapplicable to much collaborative innovation because that collaboration occurs outside the employer-employee/independent contractor context (and often outside of formal contracts). The joint work doctrine is also less helpful than it could be in reducing the number of copyright claimants, since object-oriented programming means that many of the software objects included in a larger work are self-contained, modular, and thus clearly subject to their own copyright. In sum, copyright's potentially simplifying doctrines often fail to map onto the realities of collaborative innovation models. The result is a myriad of copyright interests in any given software project.

B. Interoperability in the Software World

A related trend in software innovation that is increasingly leading to anticommons concerns lies in the growing need and ability for heterogeneous products and services to interoperate with one another.

Interoperability has long been an issue in the software world. Early on in software's history, software programs were largely written for and confined to specific hardware products.[127] Little to no interoperability thus existed. But in the decades that followed, an independent software industry developed as software vendors began selling off-the-shelf software programs meant to operate on a variety of hardware and software platforms.[128] Providing for greater interoperability among heterogeneous hardware and software products

---

125 *See generally* Mary LaFrance, *Authorship, Dominance, and the Captive Collaborator: Preserving the Rights of Joint Authors*, 50 EMORY L.J. 193 (2001) (discussing judicial efforts to narrow the definition of joint works).

126 Garcia v. Google, Inc., 786 F.3d 733, 742–43 (9th Cir. 2015).

127 *See supra* notes 58–65 and accompanying text.

128 *Id.*

thus proved to be a key selling point in this phase of software's history.[129] And, as discussed above, copyright played an important role in helping the software industry establish itself by providing a means by which software developers could recoup their costs of developing software products.[130]

But interoperability needs in the software industry have accelerated and changed, with implications for copyright's role for software more generally. First, as discussed *supra*, object-oriented programming has come to dominate the modern software industry. This model of programming stresses the development of self-contained software objects capable of invoking functionality and data from other software objects without a need for deeper integration with them.[131] Hence, while object-oriented programming does not mandate greater interoperability between software programs, it certainly set the stage for it.

Second, software has made its way into more and more everyday goods, including cars, household appliances, televisions, watches, treadmills, phones, security systems, cooling and heating systems, and more.[132] Hence, in today's world, software is not only for traditional computing devices, but instead has transformed more and more everyday goods into computing devices.

Third and related, these modern-day computing devices increasingly rely on this software to connect to other products and services. The growing Internet of Things economy, where Internet-enabled products harness Internet connectivity to interconnect with a variety of other devices and services, is one term commentators often use to refer to this trend.[133] Analysts also often refer to these Internet-enabled goods as "smart" products.[134]

---

[129] *Id.*

[130] *Id.*

[131] Smith, *supra* note 96, at 2088–89, 2088 n.16 (discussing this "modularity" as a key virtue of object-oriented programming).

[132] *See, e.g.*, Marc Andreessen, *Why Software Is Eating the World*, WALL ST. J. (Aug. 20, 2011), http://www.wsj.com/articles/SB10001424053111903480904576512250915629460 (discussing software-powered cars and the importance of software in the modern world more generally); Ry Crist, *Best Smart Home Devices of 2016*, CNET (Aug. 18, 2016), http://www.cnet.com/topics/smart-home/best-smart-home-devices/ (discussing several of these "smart" devices).

[133] John Thielens, *Without API Management, the Internet of Things Is Just a Big Thing*, WIRED http://www.wired.com/insights/2013/07/without-api-management-the-internet-of-things-is-just-a-big-thing/ (last visited Oct. 17, 2016) (describing the Internet of Things generally).

[134] *See, e.g.*, David Einstein, *Smart Devices Not a Smart Choice for Now*, SF GATE (Nov. 8, 2015, 6:54 PM), http://www.sfgate.com/business/article/Smart-devices-not-a-smart-choice-for-now-6618701.php (discussing "smart" devices generally while identifying possible problems with using such devices).

Fourth, this interconnected, smart world is facilitated through cloud computing, which allows parties to provide access to products and services through the Internet rather than solely through software installed on a user's computer.[135] For instance, in a cloud computing environment, a company hosts the software products on its own servers, but provides access to many of the benefits of that software through Internet connectivity and a web browser or, in some cases, a software app installed on a device.[136] Accessing Dropbox, whether on a laptop or smartphone, is a simple example of this type of cloud-based service. By many accounts, cloud computing is increasingly taking over as businesses and consumers shift to using technologies hosted in the cloud.[137] Hence, in addition to allowing consumers and businesses to transfer the costs of technology ownership to a third party while still obtaining the benefits of that technology, the growth of cloud computing has also made interconnecting a wide variety of heterogeneous products and services possible.

Collectively, these trends point to a number of possible benefits. Many of these benefits relate to automating aspects of everyday life, such as automatically adjusting the thermostat's temperature in a home when a homeowner unlocks her front door.[138] Indeed, consumers increasingly expect their goods and services to be capable of interfacing with each other in these ways.[139] But in order for this automation and interoperability to be possible, parties must often share and copy software code from one another, in particular software interfaces that allow for distinct programs to interoperate.[140] Hence, interoperability among the expanding universe of smart products is not

---

[135] Eric Griffith, *What Is Cloud Computing?*, PCMAG (Apr. 17, 2015), http://www.pcmag.com/article2/0,2817,2372163,00.asp (describing cloud computing generally).

[136] *Id.*

[137] *See, e.g.*, Elaina Robbins, *Top Seven Ways Cloud Computing Is Taking Over*, REVUEZZLE, http://revuezzle.com/cloud-backup/cloud-backup-articles/basics/top-seven-ways-cloud-computing-taking/ (last visited Sept. 22, 2016) (summarizing a number of independent sources that point to a major shift from traditional technologies to cloud-based ones).

[138] EAston, *The Internet of Things (IoT): Challenges and Benefits*, WT VOX (Feb. 16, 2015), https://wtvox.com/internet-of-things-iot/the-internet-of-things-iot-challenges-and-benefits/

[139] *Internet of Things: Consumer Expectations Increase with Each Smart Home Device Purchase*, PARKS ASSOCIATES (Sept. 22, 2014), https://www.parksassociates.com/blog/article/pr-sept2014-iot-webcast (finding that the importance of interoperability between products and services increases for consumers with each additional smart home device purchased).

[140] *See* Matt Schruers, *Supreme Court Declines to Hear* Oracle v. Google *Case over Java Copyrights*, DISCO PROJECT (June 29, 2015), http://www.project-disco.org/intellectual-property/062915-supreme-court-declines-to-hear-oracle-v-google-case-over-java-copyrights/ (arguing that the decision in *Oracle v. Google* will harm software innovation because it will prevent interoperability, since parties will be barred from copying Oracle's Java APIs for interoperability purposes).

possible without each participating product making available to all others some of the software technologies that power it.

An example helps illustrate these points. Amazon's "Echo" product is a home device that responds to human voice instructions to accomplish a variety of activities, including playing music, placing shopping orders, looking up information on the Internet and reading it out loud, among other possibilities.[141] The Echo thus relies on Internet connectivity to access Web-based services such as Wikipedia, Pandora, and others in order to accomplish many of its tasks. And to seamlessly interoperate with each of these third-party services, the Echo must be able to access, copy, and use certain software technologies—software interfaces—from those third parties.[142] Hence, Amazon necessarily incorporates software from those third parties within the Echo itself to enable it to effectively "speak" to those third party services.

This and many similar examples thus highlight the need to incorporate software from third parties into one's own products and services to allow the same to interoperate with the third party products through the cloud (and otherwise).[143] Often parties enter into agreements that allow for uses of each other's software technologies simply because each party sees the benefits in allowing interoperability between its products.[144] Indeed, many parties are eager for third parties to incorporate their products within their own and may, therefore, make the software necessary for interoperability readily available.[145]

---

[141] *See, e.g.*, Stacey Higginbotham, *5 Things You Can Now Do with Your Amazon Echo and 3 Things You Can't*, FORTUNE (Aug. 27, 2015, 11:36 AM), http://fortune.com/2015/08/27/amazon-echo-home-automation/.

[142] *See, e.g.*, James Kendrick, *Amazon Echo Gets Native Pandora Support, Plus MLB and MLS Information*, ZDNET (Apr. 2, 2015, 10:58 AM), http://www.zdnet.com/article/amazon-echo-gets-native-pandora-support-mlb-and-msl/ (describing the Echo device now featuring "native support" for several third-party services, which generally means in this context that the device has incorporated software from those third parties that enables the device to more seamlessly interoperate with those services).

[143] Cade Metz, *Nest Gets into the Smart-Lock Game by Going Old School*, WIRED (Oct. 1, 2015, 9:00 AM), http://www.wired.com/2015/10/nest-embraces-good-ol-yale-locks-make-smart-homes-smarter/ (discussing Nest's development of an alternative to Internet connectivity for connecting devices).

[144] *Id.* (discussing how "more than 11,000 device makers have signed up to build compatible devices" with Nest, a company that develops smart thermostats, among other devices, meant to interoperate and exchange information with third party products and services).

[145] Gary Little, *Why APIs Will Save Your Business from Getting "Uber-ed"*, FORTUNE (May 19, 2015, 12:48 PM), http://fortune.com/2015/05/19/why-apis-will-save-your-business-from-getting-uber-ed/ (discussing how UPS, among other companies, has outpaced its competition by publishing to the world "an API—just a few lines of code—which makes it easy for shopping sites to integrate shipping seamlessly into their online check-out process").

But in other cases this may not be so.[146] In fact, even in cases where a party initially sees merit in making its software interfaces available for third party use in enabling interoperability, the party may change course later. Twitter did exactly that when it terminated a deal that had allowed LinkedIn to syndicate users' tweets within LinkedIn's activity stream.[147] Twitter cited a number of business reasons for this change, including preserving an authentic Twitter experience.[148] But this turnabout nonetheless contrasts dramatically with its earlier stance as to the availability of its software technologies that enabled interoperability, which had largely allowed third parties to "do just about anything they wanted in integrating with Twitter."[149]

None of this is to say that, in many cases, Twitter and others lack good reasons for reining in their software technologies and thereby preventing interoperability with third party products and services.[150] For instance, a party may wish to retain control over its software technologies for security, privacy, business, and user experience reasons, among others.[151]

Nonetheless, the lack of standardized means by which heterogeneous products can interoperate with each other remains one of the biggest obstacles to realizing many of the advantages of interconnecting the universe of smart products.[152] And standardization cannot occur without parties allowing use of their own copyrighted software technologies—in particular software interfaces—as part of these ecosystems. This lack of standardization can thus lead to significant anticommons concerns, particularly given the presence of copyright, as described more fully below.

---

146 *Id.* (pointing to a number of rental car companies that have failed to make available software technologies that would allow third parties to seamlessly integrate their services within their own).

147 Mike Isaac, *Twitter Cuts Off LinkedIn Who's Next?*, ALL THINGS D (June 29, 2012, 6:57 PM), http://allthingsd.com/20120629/twitter-cuts-off-linkedin-whos-next/.

148 *Id.*

149 *Id.*

150 *See, e.g.*, Thielens, *supra* note 133 (discussing the importance of managing software APIs for the Internet of Things to work well and securely).

151 *Id.*

152 John F. O'Rourke & Trevor K. Roberts, *There Is No Such Thing as the Internet of Things At Least Not Yet!*, LEGALTECH NEWS (Nov. 12, 2015), http://www.legaltechnews.com/id=1202742252036/There-is-No-Such-Thing-as-the-Internet-of-Things—at-Least-Not-Yet?mcode=0&curindex=0&curpage=ALL&slreturn=20151019145317 (arguing that the Internet of Things can only come about "if the [software] protocols that will manage all of this data are standardized").

C. Anticommons in Software

In order to better understand how these two trends in software innovation lead to anticommons concerns, it is important to more fully understand what the term "anticommons" means. Michael Heller coined the now well-known phrase "tragedy of the anticommons."[153] I use "anticommons" throughout this Article as shorthand for "tragedy of the anticommons."

An anticommons is the mirror-image of a "tragedy of the commons."[154] A tragedy of the commons develops when "multiple owners are each endowed with the privilege to use a given resource, and no one has the right to exclude another."[155] When multiple parties possess unfettered ownership interests in the common resource, the resource is likely to be overused, resulting ultimately in its destruction, unless users of the resource are somehow regulated.[156] The most commonly cited examples of a tragedy of the commons are depleted fisheries and overgrazed fields.[157]

In contrast, an anticommons may arise when numerous parties have rights in a resource, but those rights fail to provide any of these parties "an effective privilege" to use the resource without permission from all the others.[158] For instance, each rights holder, on the basis of its right in the resource, can prevent the other rights holders from using the common resource.[159] But these rights of exclusion do not include a right of use, meaning that no one party can unilaterally use the resource without permission from all the others.[160]

As a matter of logic, anticommons concerns become increasingly acute as the number of parties possessing rights in a given resource increases, simply because the chances of any one of those parties using its rights in ways that inhibit use of the resource may also increase.[161] Hence, a resource may end up not being used, or used less than what is socially optimal, because rights clearance becomes increasingly cumbersome or, in some cases, impossible.

---

153 *See* Heller, *supra* note 17, at 623–24 (defining the term "tragedy of the anticommons").
154 *Id.*
155 *Id.*
156 *Id.*
157 *Id.* at 624.
158 *Id.*
159 *Id.*
160 *Id.*
161 *Id.*

The paradigmatic example of an anticommons is the image of shuttered storefronts in post-Soviet Russia.[162] The underuse (or non-use) of these properties resulted, not because parties lacked rights to them, but because none of the numerous rights holders could use the properties without permission from each of the others.[163] Hence, each property was subject to too many distinct property interests, none of which actually enabled any of the separate right holders to use the property.[164] In the world of intellectual property, scholars have pointed to biotechnology resources covered by multiple patent rights as presenting similar anticommons concerns.[165]

This Article adopts this anticommons conceptual lens in assessing copyright's effects in the software world, with some slight modifications. For instance, the paradigmatic example of an anticommons is when a rights holder refuses to allow others to use the common resource on the basis of her rights.[166] While this type of "holdup" problem may also often apply in the software context, this Article contends that the sheer number of copyright interests can also result in underuse of a resource—and thus anticommons concerns—even when rights are not aggressively asserted in this more typical fashion. The following sections assess how.

1. How Collaborative Innovation May Lead to an Anticommons

The collaborative nature of modern software innovation described above creates many of the conditions necessary for an anticommons in the software world. For instance, as discussed above, the resource in this case—any given software product—is likely to include numerous copyright interests because of the collaborative innovation models widely used in the software industry. Indeed, object-oriented programming increases this likelihood because any given software object will often depend on and reuse other software objects.[167] And each of these numerous copyright interests may give the respective copyright holder an ability to exclude others from making use of the collaboratively built resource. Hence, though many parties have copyright interests in any given collaboratively built software product, no one party has an effective privilege to use it absent permission from the numerous other

---

162 *Id.* at 622–24.
163 *Id.* at 639.
164 *Id.*
165 Heller & Eisenberg, *supra* note 17, at 698.
166 *See supra* notes 17–18 and accompanying text.
167 Song, *supra* note 120, at 689–90.

rights holders.[168] And because the collaborative nature of modern software development practices often results in any given software product including hundreds, and in some cases thousands, of separate copyright interests, clearing the rights necessary to obtain an effective privilege of use can be difficult.

Of course, as discussed above, collaborative software innovators have employed various licensing tools to facilitate their collaborative activities, which help reduce these concerns in important respects. For instance, the typical copyright licenses under which collaboratively built software projects are made available help alleviate the concern that parties will underutilize the software resources. This may be so because the licensing models in some sense ensure an "effective privilege of use" by guaranteeing access to the software resource,[169] so long as the user meets whatever conditions of use the license specifies.[170] Indeed, as others have argued, these collaborative licensing models may thus help eliminate anticommons concerns in the software context and elsewhere.[171]

While these licensing models have undoubtedly helped address some anticommons concerns, they do not eliminate them and, in fact, may create some of their own. For instance, as I and others have written, despite the generally liberal nature of these copyright licenses, the software remains subject to a multiplicity of copyright interests.[172] And those interests demand adherence to the applicable licensing terms, regardless of how onerous they may be on any given user.[173] The result is that the growing number of copyright interests in any given piece of software often leads to underuse of the

---

168 *See supra* notes 17–18 and accompanying text.

169 *Id.*

170 Robert W. Gomulkiewicz, *De-Bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75, 79–96 (2002) (reviewing some of the basics of FOSS licensing as well as highlighting some of the challenges that specific licensing requirements may pose).

171 *See supra* note 27 and accompanying text.

172 *See, e.g.*, Asay, *supra* note 22, at 759–62; Gomulkiewicz, *supra* note 170; Greg R. Vetter, *"Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 RUTGERS L.J. 53, 71–78 (2004) (laying out some of the basics of FOSS licensing).

173 Gomulkiewicz, *supra* note 170 (cataloguing various terms that may make the licensed software difficult for many potential users to actually use); Vetter, *supra* note 172, at 152–57 (laying out some of the reasons as to why certain FOSS licensing terms may be difficult for potential users to accept).

software product and thus anticommons concerns.[174] This result may be particularly so given a growing trend of aggressive license enforcement.[175]

For instance, because of the presence of copyright, parties using collaboratively built software undertake significant costs in order to ensure compliance with the applicable terms.[176] These costs include monitoring intake, use, and distribution of such software products, which can be significant and are typically an ongoing effort.[177] In some cases, parties simply ban certain types of collaboratively built software because the terms are too demanding in light of their business practices.[178] The result is reduced use of the licensed software products, and that reduced use is in part the result of a multiplicity of copyright interests that collaborative innovation models help spawn.

Indeed, the mere fact that many software copyright holders may intend for their software to be used liberally does not change this reality. In other words, even though these copyright holders may not appear to hold up use of the collaboratively built software because they do not actively assert their rights in more traditionally exclusionary ways,[179] their retention and use of copyright in achieving their objectives might be viewed as "holdup by other means" because copyright, as employed, still leads to significant exclusion. For instance, as discussed, even though these copyright holders provide that their software may be used, they nonetheless condition such use on terms that can be unpalatable for many. Indeed, in many cases, potential users would rather pay significant sums of money for use than comply with the stated terms.[180] Or

---

174 Asay, *supra* note 22, at 768; Vetter, *supra* note 172, at 152–57.

175 *See, e.g.*, Joe Fay, *VMware vs German Kernel Dev: Filings Flung in Linux-Lifting Lawsuit*, THE REGISTER (Oct. 29, 2015), http://www.theregister.co.uk/2015/10/29/hellwig_versus_vmware_update/ (detailing a lawsuit against VMware for allegedly failing to comply with such license conditions); Steven J. Vaughan-Nichols, *VMware Sued for Failure to Comply with Linux License*, ZDNET (Mar. 6, 2015, 5:26 AM), http://www.zdnet.com/article/vmware-sued-for-failure-to-comply-with-linuxs-license/ (detailing more aggressive efforts in recent years by developers of collaboratively built software to enforce license conditions).

176 Asay, *supra* note 22, at 768–75.

177 *Id.*

178 *Id.* at 770.

179 Such copyright holders do still at times assert their rights of exclusion when they perceive that users have violated the applicable license conditions. For a partial list of such lawsuits, see Heather J. Meeker, *Open Source and the Age of Enforcement*, 4 HASTINGS SCI. & TECH L.J. 267, 268–70 (2012) (providing a catalogue of FOSS-related lawsuits as of 2012).

180 This is why, for instance, some software companies use a dual license strategy for their collaboratively built software projects, in which users can choose either a proprietary license or the applicable FOSS license. *See* Philip H. Albert, *Dual Licensing: Having Your Cake and Eating It Too*, LINUXINSIDER (Nov. 16, 2004, 5:00 AM), http://www.linuxinsider.com/story/38172.html.

in other cases, though the terms may be more palatable, those multiple, often confusing terms still result in a variety of rights clearance activities that slow and impede use of the software product.[181] In both cases, numerous copyright interests and the possibility of copyright remedies thus slow adoption and use of collaboratively built software products.

Of course, it may be the case that, but for the licensing models that helped facilitate collaborative innovation in the first place, society may not have the collaboratively built software for use at all. Hence, even if the multiplicity of copyright interests leads to some underuse of the collaboratively built software, this underuse is the necessary tradeoff, the argument goes, to ensure that collaborative software innovation remains robust.[182]

I have addressed this question in other work, where I argue there is significant reason to doubt that copyright and the collaborative licensing tools based in it are responsible for ensuring that collaborative software innovation remains vigorous.[183] Indeed, other scholars have pointed to a number of factors that motivate parties to create and innovate that seem to have little to do with copyright and other intellectual property interests.[184] Furthermore, significant recent empirical evidence shows that more and more software collaborators simply contribute to collaborative projects under no copyright terms at all. For instance, studies of projects on GitHub, the dominant online open source software repository today, show that 85% of projects on the site have no copyright license assigned to them.[185] In addition, increasingly more developers are shunning licenses that impose "copyleft" terms,[186] which

---

181 Asay, *supra* note 22, at 769.

182 Moglen, *supra* note 107, at 22.

183 *See* Asay, *supra* note 22.

184 *E.g.*, Jeanne C. Fromer, *Expressive Incentives in Intellectual Property*, 98 VA. L. REV. 1745, 1771–81 (2012) (analyzing the non-pecuniary reasons that parties innovate); Sebastian V. Engelhardt, *What Economists Know About Open Source Software: Its Basic Principles and Research Results* 10–12 (Jena Econ. Research, Working Paper No. 2011–005, 2011), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1759976 (providing a literature review of relevant studies done on the motivations of programmers in contributing to open-licensed projects); Josh Lerner & Jean Tirole, *The Economics of Technology Sharing: Open Source and Beyond* 7–11 (Nat'l Bureau of Econ. Research, Working Paper No. 10956, 2004), http://www.nber.org/papers/w10956 (postulating many of the same signaling incentives that motivate software programmers while also reviewing extant surveys that confirm that such incentives do in fact motivate programmers to contribute time and resources to open-licensed projects); Josh Lerner & Jean Tirole, *The Simple Economics of Open Source* 14–19 (Harv. Bus. Sch., Working Paper No. 00–059, 2000), http://papers.ssrn.com/paper.taf?abstract_id=224008 (discussing the "signaling incentives" that motivate software programmers to participate in FOSS projects).

185 Neil McAllister, *Study: Most Projects on GitHub Not Open Source Licensed*, THE REGISTER (Apr. 18, 2013, 1:22 PM), http://www.theregister.co.uk/2013/04/18/github_licensing_study/.

186 *Id.*

founders of the open source software movement have argued are crucial to ensuring robust collaboration in the software commons.[187] Hence, this evidence suggests the underuse of collaboratively built software resources that reliance on copyright leads to may not actually be needed.

Even assuming that some underuse of collaboratively built software is a necessary tradeoff for otherwise robust software collaboration—which this Article and prior scholarship dispute—it is a tradeoff that has been largely overlooked or even assumed to be negligible.[188] But as this Article argues, the collaborative nature of modern software development results in significant anticommons potential because modern software products consist of increasingly more copyright interests. And any one of those copyright interests may ultimately translate into underuse of the software resource because of the way in which copyright is deployed.

2. *How Growing Interoperability Needs May Lead to an Anticommons*

The interoperability necessary to realize many of the benefits of the interconnected economy also creates significant anticommons concerns. In the interoperability context, the resource that may be underutilized—and thus subject to anticommons concerns—is the universe of things that might otherwise interoperate with each other, rather than the individual software resources. As discussed above, this universe is quickly expanding as more and more goods use software to function. Yet the possibility of these goods and services interconnecting in socially beneficial ways only becomes a reality when the software technologies necessary for such interoperability—primarily software interfaces—are available for use. Hence, the vast possibilities associated with this resource depend in significant part on parties opting to participate in it by making these software technologies available to others.

Copyright may decrease the chances of this happening. For instance, if software interfaces are subject to copyright, the developer of any given set of interfaces may withhold permission to use these technologies and thereby prevent the benefits of an interconnected ecosystem of devices and services. And because these interconnected ecosystems (ideally) implicate thousands of

---

[187] Moglen, *supra* note 107, at 19–23.

[188] *Id.* (trumpeting the success of the free software movement and failing to acknowledge any possible unintended consequences—such as underuse of the software resources—that might result from the licensing terms applied to such resources). For a study that does assess such possibilities, see Vetter, *supra* note 172, at 144–56 (discussing whether "infectious" licensing terms may actually lead to underuse of software resources rather than facilitating use and further development of them).

separate parties with distinct copyright interests, anticommons concerns grow simply because there are more chances that some parties will, in fact, use copyright in ways that prevent such interoperability. Hence, if software interfaces and other technologies necessary to enable interconnectedness are subject to copyright, the possibility of such interconnectedness decreases.

Of course, concerns about copyright holders using their rights to prevent interoperability are not new. In the past, some circuit courts have addressed these concerns by allowing for reuse of the software technologies necessary for interoperability.[189] For instance, the Sixth Circuit denied copyright relief to Lexmark, a manufacturer of printers and related printer toner cartridges, when Lexmark sought to prevent competitors from using some of its software technologies necessary to supply compatible printer toner cartridges for Lexmark's printers.[190] Similarly, the Ninth Circuit ruled against Sega when that company attempted to use copyright to prevent a third-party game manufacturer from using some of Sega's software technologies to produce games compatible with Sega's gaming console.[191]

But in today's world, where interoperability needs are growing, these precedents may prove insufficient. In these earlier cases, for instance, the copyright question focused on bilateral interoperability between the products of two parties. In today's world, however, multilateral interoperability is often the more relevant question. For instance, can party A reuse some of party B's software technologies, not simply to increase compatibility with party B's products, but in order to enhance interoperability with the larger universe of potentially interconnected things? As will be discussed further in section D, the leading modern appellate case to address this issue, *Oracle v. Google*, answered this question in the negative,[192] although on retrial the district court jury found Google's use of Oracle's software interfaces to be a fair use.[193] That

---

189 *E.g.*, Lexmark Int'l, Inc. v. Static Control Components, Inc., 387 F.3d 522, 529–31, 533–45 (6th Cir. 2004); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1993); *see also* Matthew Sag, *Copyright and Copy-Reliant Technology*, 103 NW. U. L. REV. 1607, 1624–56 (2009) (pointing to a thread of case law that appears to reject copyright claims where the copying was done for a non-expressive purpose); Matthew Sag, *Orphan Works as Grist for the Data Mill*, 27 BERKELEY TECH. L.J. 1503, 1528–42 (2012).

190 *Lexmark*, 387 F.3d at 529–31, 533–45

191 *Sega*, 977 F.2d at 1527–28.

192 Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1353–54 (Fed. Cir. 2014); *see generally* Pamela Samuelson, *Three Fundamental Flaws in CAFC's* Oracle v. Google *Decision*, 37 EUR. INTELL. PROP. REV. 702 (2015) (reviewing the *Oracle v. Google* decision and noting problems with the outcome).

193 Joe Mullin, *Google Beats Oracle Android Makes "Fair Use" of Java APIs*, ARSTECHNICA (May 26, 2016, 4:03 PM), http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/.

decision, of course, is likely to be appealed.[194] Furthermore, other circuits have not explicitly wrestled with this broader interoperability question, although many of the same copyright factors discussed in the bilateral context—such as copyright's idea-expression, scènes à faire, and merger doctrines—will certainly be relevant in answering the question in the multilateral one.[195] Hence, interoperability rationales for limiting the reach of software copyright may need a modern reboot in order to help stem an anticommons in an increasingly interconnected world.

It should be noted that copyright is not the only factor that may stand in the way of an interconnected world of things. For starters, too seamless of integration between heterogeneous products and services raises a number of privacy and security concerns.[196] Too smart of a home may mean that hackers have an easier time infiltrating the most intimate, sensitive parts of our lives, for instance.[197] Relatedly, companies may have legitimate reasons, such as user experience concerns, for wanting to prevent integration of their goods and services with those of third parties.[198]

Indeed, companies can employ a variety of means besides copyright to prevent their goods and services from becoming integrated with those of other parties; the simplest means may be to build the good or service so that it is not capable of interoperating with third-party products. Furthermore, as the Twitter example demonstrates, companies often use technical and contractual mechanisms to prevent others from integrating third-party goods and services with theirs.[199]

But despite these other possible roadblocks to an interconnected world of things, copyright stands out as particularly problematic for a number of reasons. First, allowing copyright to stand in the way of interoperability arguably does not serve the purposes of copyright. Copyright, after all, is a

---

194 *Id.*

195 *E.g.*, *Lexmark*, 387 F.3d at 534–37 (discussing copyright's idea-expression, scènes à faire, and merger doctrines, which are relevant to the broader interoperability question).

196 *See* Kenneth Corbin, *Congress Probes Internet of Things Privacy, Security*, CIO (July 30, 2015, 12:22 PM), http://www.cio.com/article/2954531/government/congress-probes-internet-of-things-privacy-security.html (discussing some of the privacy and security concerns associated with the Internet of Things).

197 Pierluigi Paganini, *How Hackers Violate Privacy and Security of the Smart Home*, INFOSEC INSTITUTE (Sept. 11, 2015), http://resources.infosecinstitute.com/how-hackers-violate-privacy-and-security-of-the-smart-home/.

198 *See* Isaac, *supra* note 147 (highlighting an example where Twitter limited access to its service by LinkedIn).

199 *Id.*

body of law authorized under the Constitution to promote the progress of "Science and useful Arts."[200] Under the dominant theoretical understanding of this constitutional provision, copyright is meant to provide authors with sufficient incentives to create original works of authorship.[201] But in situations where copyright is employed to prevent interoperability, copyright often seems to be serving other interests, such as protecting privacy, security, or in some cases simply inhibiting competition.[202] And this is problematic in part because copyright law rights, limitations, infringement standards, defenses, and remedies have all been designed with a different set of interests in mind.[203]

None of this is to say that privacy, security, and competition concerns are without merit. But it is to say that copyright, given its purpose and how that purpose is reflected in copyright law's many doctrines, is typically not the appropriate tool for addressing those concerns.[204] Copyright may, in a sense, bolster the technical and contractual measures that parties use to protect these interests, and thus help safeguard the same concerns, by providing a party with powerful copyright remedies.[205] But using copyright in this way also creates greater risks of underutilizing the resource—and thus anticommons problems—because of the deterrent effect of copyright remedies.

Indeed, as can be seen in the *Lexmark* and *Sega* cases mentioned above, copyright holders may frequently seek to use copyright to protect interests that lie outside of copyright's purposes. In fact, others have noted a growing trend of parties attempting to use copyright to protect interests unrelated to copyright's purposes, such as when doctors use copyright to quash bad online

---

200 U.S. CONST. art. I, § 8, cl. 8.

201 *See* Landes & Posner, *supra* note 33, at 325–33 (articulating the traditional view that copyright law is justified as a corrective to market failure, in which copyright interests are granted to parties as incentives to engage in creative behavior in order to overcome the market failure).

202 *Cf.* Clark D. Asay, *Ex Post Incentives and IP in* Garcia v. Google *and Beyond*, 67 STAN. L. REV. ONLINE 37, 38, 40–44 (2014) (articulating concerns about the use of copyright to protect non-copyright interests in other contexts); Jeanne C. Fromer, *Should the Law Care Why Intellectual Property Rights Have Been Asserted?*, 53 HOUS. L. REV. 549, 587–93 (2015) (arguing generally that courts should care why intellectual property rights are asserted and strongly consider denying relief in situations where the purposes behind the assertion do not align with the body of law's purpose).

203 *See* Rebecca Tushnet, *How Many Wrongs Make a Copyright?*, 98 MINN. L. REV. 2346, 2361–74 (2014) (noting a variety of problems with expanding copyright to protect interests outside of incentivizing parties to create, the traditional rationale of granting copyright).

204 *Cf.* Fromer, *supra* note 202, at 553–75; Tushnet, *supra* note 203, at 2361–74.

205 *See* 17 U.S.C. §§ 502, 504 (2012) (providing for the remedies of injunctive relief and statutory damages under copyright law).

reviews[206] or embarrassed newscasters resort to copyright to prevent dissemination of newscasts that went awry.[207] Hence, without reining in copyright for software technologies, parties in the interconnected world may increasingly use copyright similarly in protecting interests for which copyright was not intended. And when they do, anticommons concerns grow because that interconnected world becomes less connected than it otherwise might be.

A second, related reason that copyright stands out as particularly problematic is, simply, that copyright makes the standardization of software technologies for interoperability less likely. As mentioned, the lack of standardization in these types of technologies is one of the biggest obstacles to an interconnected world. Copyright makes this standardization more difficult because if different parties' software interoperability technologies are subject to copyright, the chances of collaboration and reuse of those technologies decrease. For instance, if copyright forces each party to develop its own, distinct set of software technologies for interconnecting with other products, then interconnecting the universe of products becomes more difficult. With standardization, conversely, ultimately interconnecting a variety of products and services becomes more likely.

Another way to think about this same point is to view the software technologies necessary for interoperability as a vocabulary. If disparate parties use a common vocabulary in how their products interact with third-party goods and services, then making that universe of things compatible is a much more tractable problem. If each product or service is forced to have a distinct vocabulary, on the other hand, then interconnecting that universe of things becomes a much more difficult task.

Of course, parties could simply coordinate with each other and decide upon common software protocols for interoperability, similar to what often happens with respect to patent rights and standards bodies.[208] In fact, this transactional approach to the problem aligns in some respects with certain theoretical views.

---

[206] Graeme McMillan, *Doctors Now Using Breach of Copyright to Quash Bad Online Reviews*, TIME: TECHLAND BLOG (Apr. 14, 2011), http://techland.time.com/2011/04/14/how-do-doctors-avoid-bad-online-reviews-legally/.

[207] Kristin Bergman, *After On-Air Mishaps, Embarrassed Newscasters Turn to Copyright Law*, DIGITAL MEDIA LAW PROJECT (Aug. 13, 2013, 3:41 PM), http://www.dmlp.org/blog/2013/after-air-mishaps-embarrassed-newscasters-turn-copyright-law.

[208] *See generally* Mark A. Lemley, *Intellectual Property Rights and Standard-Setting Organizations*, 90 CAL. L. REV. 1889 (2002) (discussing the importance of standard-setting organizations in governing patent rights).

Put simply, one perspective is that copyright actually enables the interconnected economy, rather than undermines it, by granting parties rights in their software technologies, which rights then allow parties to more effectively transact with one another in protecting their interests while simultaneously promoting those of the public.[209] Hence, copyright holders get some return for their investment in creating the software technologies, while the public benefits from the development of an interconnected economy.

While this argument has some theoretical appeal, it takes the property rights as a given. In other words, this approach's primary point is that property rights—and the remedies associated with them—facilitate beneficial transactions because parties are emboldened to pursue these transactions, knowing that they need not rely solely on contractual remedies in the event of a transaction falling apart.[210] But putting these transactional benefits to the side for a moment, it is vital to ask whether the property right was justified or needed in the first place? Indeed, while property rights in some cases may facilitate transactions that result in beneficial relationships between parties, the lack of property rights undoubtedly facilitates greater use of the resource, assuming that the creator of that resource was willing to develop it absent property rights. Hence, it cannot be said that property rights are warranted in every instance simply because they may facilitate transactions, particularly when ultimately unnecessary transactions over time may result in an expansion of rights that create additional anticommons concerns.[211]

With respect to software technologies necessary for interoperability, several reasons suggest that parties would still have incentives to create them even absent property rights. First, today's economy increasingly demands such interconnectedness,[212] and parties that fail to satisfy those demands may simply fail commercially. These demands do not mean that any given party will always have incentives to allow others to use their software technologies in order, for instance, to make a competitive product with interoperability capabilities. But it does generally mean that parties will have incentives to

---

[209] *See* Robert P. Merges, *A Transactional View of Property Rights*, 20 BERKELEY TECH. L.J. 1477, 1519–20 (2005) (highlighting the use of copyright as an effective tool in creating enforcement options for the contracting parties).

[210] *Id.*

[211] *See generally* James Gibson, *Risk Aversion and Rights Accretion in Intellectual Property Law*, 116 YALE L.J. 882 (2007) (discussing the "doctrinal feedback" created by copyright law and those who make use of it where the prudent practice of securing unnecessary copyright licenses expands the reach of copyright entitlements).

[212] *See supra* note 139.

continue to develop software interfaces and to collaborate with others in interconnecting their goods and services.

Second, in cases where parties wish to participate in the interconnected economy but also desire to protect privacy, security, user experience, or other competitive interests, means other than copyright, such as contract and technological solutions, are available to help address these concerns. Of course, these solutions, absent copyright and its associated remedies, may not be as robust. For instance, if some party reverse engineers and then copies a technological mechanism in a product meant to safeguard the privacy of the product's users, then the absence of copyright remedies may mean that those privacy interests are less capable of vindication. And that possibility may mean that a party is less incentivized to create the product. But, again, making copyright the vindicator of all possible interests is the wrong approach for a number of reasons discussed above.[213] While copyright may be a convenient solution for such problems, that convenience masks deeper problems in using copyright as a panacea for all possible issues.

Last, while property rights may make sense for software more broadly, they seem less crucial with respect to the more limited set of software technologies that simply allow heterogeneous products and services to speak with one another. While designing these technologies certainly requires effort, effort alone is generally not considered a basis for copyright protection.[214] It is certainly true that a party may wish to prevent others from copying such technologies in creating competitive products. But, as courts and scholars have long recognized, preventing such competition by granting rights in these functional software elements generally falls outside the purposes of copyright law.[215]

Instead, the primary result of subjecting interoperability technologies to strong copyright protections may be a growing anticommons in an interconnected world. The next section examines Google's Android and a copyright controversy surrounding it in order to provide one recent, high-profile example of how copyright may lead to significant anticommons

---

[213] *See supra* notes 200–07 and accompanying text.

[214] *See* Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 359–61 (1991) (rejecting a "sweat of the brow" rationale for granting copyright protection).

[215] *See* Samuelson, *supra* note 81, at 73–86; Samuelson, *supra* note 79; *supra* note 189 and accompanying text.

concerns because of the collaborative, interconnected nature of modern software innovation.

D. Android's Java Problem

Google's Android operating system for mobile devices is a collaboratively built software platform. While Google takes the lead in engineering and releasing new versions of the software platform, the project is made available under a variety of open source software licensing terms.[216] This licensing scheme means that any third party can take and use Android as it wishes, so long as it complies with the relevant licensing conditions.[217] Amazon, for instance, has done exactly that in using Android in its various mobile devices.[218] These licensing terms also mean that a variety of third parties can and do contribute to the official version of Android that Google maintains and releases.[219] Furthermore, Android was built using object-oriented programming, meaning it includes a variety of software objects from third-parties.[220] As a result, Android contains copyrighted contributions from numerous third parties, as well as materials that Google has incorporated into the project itself.[221]

Some of the materials that Google incorporated into Android include thirty-seven of Oracle's Java APIs. As discussed throughout, APIs can generally be understood as software technologies that make it possible for different software programs to interact with each other and share data, without having to otherwise rewrite each individual program.[222] APIs are incredibly important; they enable many things that computer users take for granted, such as being

---

[216] For a general overview of Android, see Hildenbrand, *supra* note 123.

[217] *See Licenses*, ANDROID, https://source.android.com/source/licenses.html (last visited Oct. 18, 2016) (providing an overview of the terms under which Android is made available to the public).

[218] Greg Lamm, *How Amazon Uses Android for Kindle Fire, but Cuts Google Out*, PUGET SOUND BUS. J. (Jan. 24, 2012, 8:22 AM), http://www.bizjournals.com/seattle/blog/techflash/2012/01/how-amazon-uses-andorid-for-kindle-fire.html.

[219] *See, e.g.*, *Contributing*, ANDROID, https://source.android.com/source/contributing.html (last visited Oct. 18, 2016) (encouraging external parties to contribute software to the Android project). Of course, Google ultimately decides which contributions make it into the official version of Android. *Id.*

[220] *See generally Android Architecture*, EASYTUTZ (Feb. 3, 2015), http://www.eazytutz.com/android/android-architecture/ (describing Android's architecture in detail).

[221] *See, e.g.*, *Google Individual Contributor License Agreement*, GOOGLE DEVELOPERS, https://cla.developers.google.com/about/google-individual (last visited Oct. 18, 2016) (making clear that contributors retain copyright ownership in their contributions, while granting permissive licenses to other developers and users of the Android project).

[222] Brian Proffitt, *What APIs Are and Why They're Important*, READWRITE (Sept. 19, 2013), http://readwrite.com/2013/09/19/api-defined.

able to copy text from one application to another or having the Yelp app display nearby restaurants on a Google Map in the app.[223]

When Google elected to incorporate the Java APIs into Android, it did so for a number of reasons that are relevant to this Article's inquiry.[224] First, the Java APIs had become an industry standard; programmers were accustomed to creating their applications using the Java programming language and incorporating these APIs in order to perform the types of computing functions that the APIs specify.[225] Hence, though Google could have developed their own nomenclature for similar computing functions, doing so would have been inefficient, both from the company's perspective and that of third party programmers.[226] For instance, programmers would have had to completely rewrite many of their software programs in order to make them work properly with Android.[227] And even if third party programmers did write applications from scratch for use with Android, their preexisting familiarity with the Java APIs made choosing those APIs for Android a natural choice for Google.[228] In other words, the interconnectedness of software innovation in today's world made incorporating the APIs into Android imperative.

Second, in addition to these efficiency considerations, Java technologies also presented a number of technical merits when compared to other options.[229] Sun Microsystems had built the Java technologies with the intention to make it simpler for them to seamlessly work across a variety of heterogeneous software and hardware platforms.[230] Thus, despite the Java solutions having a number of technical shortcomings in this regard (which Google sought to overcome in Android), the basic design of Java technologies still presented significant technical merits in fostering collaboration and interoperability. Furthermore, Sun released much of the Java technology to the public under

---

223 *Id.*

224 Stephen Shankland, *Android, Java, and the Tech Behind* Oracle v. Google *(FAQ)*, CNET (Apr. 20, 2012, 4:00 AM), http://www.cnet.com/news/android-java-and-the-tech-behind-oracle-v-google-faq/ (laying out the history of Google's adoption of Java technology as part of Android and providing many of the rationales for this adoption).

225 *Id.*

226 *Id.*

227 *Id.*

228 Indeed, as one senior engineer at Google at the time of Google's decision to use Java for Android said, all the alternatives to Java "sucked." Richard Waters, *Android Boss on the Alternatives to Licensing Java: "They All Suck"*, FINANCIAL TIMES: TECH BLOG (Aug. 3, 2011, 6:16 PM), http://blogs.ft.com/tech-blog/2011/08/android-boss-on-the-alternatives-to-licensing-java-they-all-suck/.

229 *Id.*; *see also* Shankland, *supra* note 224.

230 Shankland, *supra* note 224.

open source software licensing terms, thereby encouraging third parties to adopt and further develop these technologies.[231] Collaboration and interoperability were thus important foundations of the Java technologies. And these foundations were important considerations when Google elected to incorporate the Java APIs into Android.[232]

The story of Android shows how the collaborative, interconnected nature of much software development leads to many of the anticommons concerns discussed above. For instance, parties wishing to use Android (or pieces thereof) must undertake significant efforts in reviewing the relevant copyright licenses and complying with the terms thereof. As briefly mentioned above, these types of compliance efforts are pervasive and relate not only to the use of Android, but to collaboratively built projects more generally.[233] But Android is no ordinary project—its ten million lines of code mean that potential users must take into account thousands of separate copyright interests subject to a variety of different license requirements.[234] The end result is reduced—or at least slowed—adoption of different pieces of the project as potential users grapple with this vast sea of copyright interests.

Indeed, while standardization of the relevant licenses and compliance automation tools may help ease this burden, the reality is that parties cannot fully rely on such standardization and automation if they wish to fully comply with all relevant copyright licenses.[235] And while a general spirit of liberality

---

[231] China Martens, *Sun Open Sources Java Under GPL*, INFOWORLD (Nov. 13, 2006), http://www.infoworld.com/article/2660378/application-development/sun-open-sources-java-under-gpl.html (discussing Sun's decision to make several core Java technologies available under permissive licensing terms).

[232] Shankland, *supra* note 224.

[233] *See, e.g.*, Michael Dolan, *Why Companies that Use Open Source Need a Compliance Program*, LINUX.COM (June 1, 2015), https://www.linux.com/news/featured-blogs/205-mike-dolan/833369-why-companies-that-use-open-source-need-a-compliance-program (discussing the rising use of collaboratively built software and the need for companies making such use to ensure compliance). Indeed, prominent lawyers in this field of law have written books on best practices relating to compliance. *See generally* HEATHER MEEKER, OPEN (SOURCE) FOR BUSINESS: A PRACTICAL GUIDE TO OPEN SOURCE SOFTWARE LICENSING (2015) (providing nearly 300 pages of compliance tips for businesses intending to use collaboratively built software). Furthermore, companies exist whose primary business consists of helping users of collaboratively built software comply with the relevant licensing terms. *E.g.*, *Open Source License Compliance & Governance*, BLACK DUCK, https://www.blackducksoftware.com/solutions/compliance (last visited Oct. 18, 2016).

[234] *See* Lee, *supra* note 15.

[235] *See* Ibrahim Haddad, *7 Steps to Strengthen Your Open Source Compliance*, SAMSUNG OPEN SOURCE GROUP (May 4, 2015), http://blogs.s-osg.org/7-steps-to-strengthen-your-open-source-compliance/ (providing a detailed look at best practices in ensuring compliance with open source software licensing terms, some of which rely on growing automation, but many of which can never be fully automated). For an example of a full-length book dedicated to explaining how best to address compliance issues relating to use of open source software, see Meeker, *supra* note 233.

relating to the project may help assuage fears of copyright infringement allegations, the reality is that many collaborative developers are growing more assertive in enforcing their copyright interests against those they believe do not strictly comply with the relevant licensing terms.[236]

Of course, another view is that these potential anticommons concerns pale in comparison to the benefits of the software commons that collaborative licensing models have helped build, as discussed above. Hence, copyright, whatever problems it entails, may actually ensure the success of collaborative software innovation.[237] It does so in some cases, for instance, by requiring that users of the collaboratively built software make their improvements to such software available to the public under the same liberal terms.[238] But Android is not good evidence in support of this argument, since much of Android is subject to licensing terms that make no such requirement.[239] Furthermore, as I have argued above[240] and elsewhere,[241] it is increasingly doubtful that copyright actually plays these purported roles effectively, particularly in light of the growing anticommons concerns described in this Article.

The recent *Oracle v. Google* decision may exacerbate these anticommons concerns. As briefly mentioned *supra*, in *Oracle v. Google* the Court of Appeals for the Federal Circuit overturned the district court's decision, ruling that the Java APIs were subject to copyright and remanding the case to the district court for a fuller consideration of Google's fair use arguments.[242] Google filed a petition for writ of certiorari with the Supreme Court in 2015, but its request was denied.[243] On remand, the district court jury found that

---

[236] For instance, the Software Freedom Conservancy launched the "GPL Compliance Project for Linux Developers" project in May 2012 as an effort to help developers enforce their copyrights against parties failing to comply with the applicable license terms. *See Conservancy's Copyleft Compliance Projects*, SOFTWARE FREEDOM CONSERVANCY, https://sfconservancy.org/linux-compliance/ (last visited Nov. 9, 2015); *see also supra* note 179 and accompanying text. Of note, Android includes parts of Linux. *Android is Based on Linux, but What Does That Mean?*, HOW-TO-GEEK, http://www.howtogeek.com/189036/android-is-based-on-linux-but-what-does-that-mean/ (last visited Oct. 18, 2016).

[237] Moglen, *supra* note 107, at 21–23.

[238] *Id.* at 20–23.

[239] Much of Android is subject to the Apache 2.0 license, which mainly requires that users of the code include in their own distributions of the same code relevant copyright notices and a copy of the Apache license. *See Licenses*, ANDROID, https://source.android.com/source/licenses.html (last visited Dec. 8, 2015).

[240] *See* notes 182–87 and accompanying text.

[241] *Supra* note 22 and accompanying text.

[242] Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014).

[243] Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887, 2887 (2015).

Google's use of the Java APIs was a fair use.[244] Nonetheless, the decision is likely to be appealed, leaving some ongoing legal uncertainty.[245] Furthermore, as discussed above, other circuits have not addressed the precise issue in *Oracle v. Google*. Hence, this continuing lack of legal certainty increases anticommmons concerns because it provides any number of parties whose software technologies are found in Android with a colorable legal claim against Google, users of Android, and developers of Android.[246]

III. ADDRESSING ANTICOMMONS CONCERNS IN THE SOFTWARE WORLD (AND ELSEWHERE)

The previous Part argued that the collaborative, interconnected nature of much modern software innovation means that anticommons concerns will increasingly bedevil the software industry and information technology sector more generally. For instance, these characteristics of modern software innovation mean, simply put, that numerous copyright interests may be present in any given software solution or ecosystem, which increase the risks of any given copyright holder asserting or using their rights in ways that inhibit collective use of the resource. Part II also briefly examined Oracle's copyright suit against Google's Android as one recent, high-profile example of some of these anticommons concerns at play. The question then becomes: what, if anything, can be done to help alleviate these issues?

This Part explores one potential answer to that question. It assesses the merits of adapting copyright law's fair use defense to the collaborative, interconnected realities of much modern software innovation. The following sections discuss how these adaptations may be made.

Before proceeding to that analysis, however, it should be noted that other options for addressing these issues exist. For instance, scholars have long argued that software interfaces, because of their functional nature, should not be subject to copyright at all.[247] And several courts have ruled along these lines in the past, including the district court in the *Oracle v. Google* case (before the

---

[244] Kate Conger, *Jury Finds Google's Implementation of Java in Android Was Fair Use*, TECHCRUNCH (May 26, 2016), http://techcrunch.com/2016/05/26/jury-finds-googles-implementation-of-java-in-android-was-fair-use/.

[245] Mullin, *supra* note 193.

[246] Pamela Samuelson, *Why Google's Fair Use Victory over Oracle Matters*, THE GUARDIAN (May 31, 2016, 5:25 PM), https://www.theguardian.com/technology/2016/may/31/google-fair-use-victory-oracle-software-androids (describing the types of parties that may be liable had Oracle won the fair use trial).

[247] Samuelson, *supra* note 79, at 14–16.

Federal Circuit ultimately overturned its decision).[248] In general, I agree with the reasoning of these scholars and courts. But I still assess fair use as a possible solution since fair use is the relevant inquiry in ongoing cases, such as *Oracle v. Google*, and is likely to be the relevant inquiry in future cases as well. Furthermore, these scholars and courts have limited their analyses to whether software interfaces should be subject to copyright, while this Article's analysis is broader. In other words, some of the software in today's collaborative, interconnected software industry is and clearly should be subject to copyright, but fair use may still play a role in allowing for its use without infringement liability.

Another option for dealing with these types of anticommons issues is a compulsory license regime, similar to what exists in the world of music.[249] A compulsory licensing regime in the software world would mean that third parties could use the copyrighted software so long as they paid a fee, either set by regulators[250] or a collective rights organization.[251] Indeed, some legal theory suggests that addressing anticommons problems through these types of liability rules is the correct approach.[252]

But even if this is theoretically true, the practical and political hurdles in implementing a compulsory licensing regime are significant. For instance, compulsory licensing regimes in other areas such as music are plagued with problems of their own, and many of these same problems are likely to surface in the software context as well.[253] Fair use, on the other hand, provides a flexible tool that is well-suited to addressing instances of market failure.[254]

---

[248] *See* Corynne McSherry, *Dangerous Decision in* Oracle v. Google*: Federal Circuit Reverses Sensible Lower Court Ruling on APIs*, ELEC. FRONTIER FOUND. (May 9, 2014), https://www.eff.org/deeplinks/2014/05/dangerous-ruling-oracle-v-google-federal-circuit-reverses-sensible-lower-court.

[249] For a brief but excellent overview of some music copyright basics, see Kristelia A. Garcia, *Facilitating Competition by Remedial Regulation*, 31 BERKELEY TECH. L.J. 183 (2016).

[250] *Id.* at 192–96.

[251] For a discussion of how these collective rights organizations operate as public-private hybrid solutions to transaction-cost problems, see Peter DiCola & Matthew Sag, *An Information-Gathering Approach to Copyright Policy*, 34 CARDOZO L. REV. 173, 208–09 (2012).

[252] *See* Guido Calabresi & A. Douglas Melamed, *Property Rules, Liability Rules, and Inalienability: One View of the Cathedral*, 85 HARV. L. REV. 1089, 1127 (1972).

[253] *See, e.g.*, Jane C. Ginsburg, *Copyright and Control over New Technologies of Dissemination*, 101 COLUM. L. REV. 1613, 1642–45 (2001) (describing problems with statutory licensing schemes); Mark A. Lemley, *Dealing with Overlapping Copyrights on the Internet*, 22 U. DAYTON L. REV. 547, 583 (1997) (noting the complexity of compulsory licensing schemes); Robert P. Merges, *Contracting into Liability Rules: Intellectual Property Rights and Collective Rights Organizations*, 84 CAL. L. REV. 1293, 1308–16 (1996) (providing critiques of compulsory licensing regimes).

[254] *See supra* note 43 and accompanying text.

And while many have lamented its unpredictability, others have shown that fair use is not so uncertain as claimed, instead highlighting it as an increasingly important tool in helping solve many important modern copyright questions.[255] The following sections assess how fair use may be adapted to help address one such copyright dilemma: growing anticommons concerns in today's software world.

A. Rebooting Fair Use

Fair use has long been a primary means of permitting socially beneficial uses of copyrighted materials that, absent the defense, would infringe a copyright holder's rights.[256] Google, for instance, successfully relied on the defense in copying and digitizing, without permission, millions of books for use in its Google Books project.[257] Fair use has also played a significant role in allowing important technologies such as VCRs[258] and Internet search engines.[259]

Over the years, fair use has also grown in importance in the software context.[260] In *Sega v. Accolade*, for instance, the Ninth Circuit found Accolade's copying and disassembling of Sega's software in order to make compatible games for Sega's gaming consoles to be a fair use.[261] Subsequent

---

[255] Tushnet, *supra* note 44, at 871 ("Critics charge that fair use is unpredictable and inconsistent with the rest of copyright law, but—like many a building material—a doctrine can be both flexible and also strong enough to support reliance."); *see also* Michael J. Madison, *A Pattern-Oriented Approach to Fair Use*, 45 WM. & MARY L. REV. 1525, 1528–30 (2004); Neil Weinstock Netanel, *Making Sense of Fair Use*, 15 LEWIS & CLARK L. REV. 715, 718 (2011); Matthew Sag, *Predicting Fair Use*, 73 OHIO ST. L.J. 47 (2012); Pamela Samuelson, *Unbundling Fair Uses*, 77 FORDHAM L. REV. 2537, 2541 (2009).

[256] *See* Lydia Pallas Loren, *Fair Use: An Affirmative Defense?*, 90 WASH. L. REV. 685, 686 (2015) ("No one doubts that the fair use doctrine is a critically important part of U.S. copyright law" because it "provides a guarantee of 'breathing space within the confines of copyright'" (quoting Campbell v. Acuff-Rose Music, 510 U.S. 569, 579 (1994))).

[257] Robinson Meyer, *After 10 Years, Google Books Is Legal*, THE ATLANTIC (Oct. 20, 2015), http://www.theatlantic.com/technology/archive/2015/10/fair-use-transformative-leval-google-books/411058/.

[258] Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 454–56, (1984) (denying a claim of contributory copyright infringement because VCRs are capable of substantial non-infringing uses such as time-shifting home videos, which was deemed fair use of the copyrighted material).

[259] Perfect 10 v. Amazon.com, Inc., 508 F.3d 1146, 1168 (9th Cir. 2007) (finding Google's display of thumbnail versions of copyrighted material in its search results constituted fair use).

[260] *See, e.g.*, Stephen M. McJohn, *Fair Use of Copyrighted Software*, 28 RUTGERS L.J 593 (1997) (examining Supreme Court cases where fair use played a key role and proposing additional modifications to the doctrine so as to foster creativity); Samuelson, *supra* note 81 (reviewing some of the early software-related cases establishing how fair use applies to use of copyrighted software).

[261] 977 F.2d 1510, 1518, 1524–25 (9th Cir. 1993).

cases have similarly relied on fair use in determining questions of copyright infringement with respect to software goods and services.[262]

Yet despite these cases, a number of scholars have expressed concern that the fair use defense is not adequately tailored to address modern technological environments.[263] For instance, the fair use test consists of four non-exhaustive factors,[264] yet none of those factors explicitly takes into account how use of the copyrighted work may impact technological innovation.[265] Application of traditional fair use principles may thus prove insufficient in rendering decisions that promote, rather than hinder, modern technological innovation.

Such a concern is particularly poignant in the software world because of software's technological, utilitarian nature. Indeed, because software is itself a type of technology, questions about its uses will frequently concern technological considerations. Yet because the fair use inquiry on its face is agnostic to technological considerations, courts may often apply the fair use factors in software cases in ways that ignore, or at least give insufficient attention to, such considerations.

Furthermore, as discussed in Part II, software innovation in today's world is increasingly collaborative and interconnected. Yet traditional fair use principles are not specifically geared towards these new technological realities either. Fair use may thus increasingly fail to strike a productive balance among the multitude of creators in today's collaborative, interconnected software world.

For instance, though a long line of software cases has applied fair use in a way that allows for use of copyrighted software to promote interoperability,[266] the fair use factors on their face do not dictate this outcome. And in light of cases such as the Federal Circuit's decision in *Oracle v. Google*, which seemed to downplay the role of interoperability in assessing copyright questions more

---

[262] *See* Samuelson, *supra* note 255, at 2605–10 (reviewing case law subsequent to the *Sega* case applying fair use principles to software-related copyright disputes).

[263] *See, e.g.*, Asay, *supra* note 49 (2016) (arguing that the fair use defense should incorporate principles from patent law to better promote technological innovation); Edward Lee, *Technological Fair Use*, 83 S. CAL. L. REV. 797 (2010) (arguing for better adapting the fair use defense to situations where speech technologies are used in conjunction with copyrighted materials).

[264] 17 U.S.C. § 107 (2012) (laying out the fair use factors).

[265] Lee, *supra* note 263, at 798–801.

[266] *See, e.g.*, Pamela Samuelson, *supra* note 79 (tracing the evolution of intellectual property law protection for software application programming interfaces).

generally,[267] there are grounds for concern that the growing interoperability needs of modern software developers will receive insufficient attention in resolving future copyright disputes.

This Article thus suggests that interoperability needs, as well as the collaborative nature of much modern software innovation, deserve greater consideration when courts assess fair use in the software context. In order to show how courts could structure the fair use inquiry accordingly, this Article dissects fair use into its component parts and analyzes each separately.

A fair use inquiry typically involves assessing four non-exhaustive statutory factors: (1) the purpose and character of the use, (2) the nature of the copyrighted work, (3) the amount and substantiality of the copyrighted work used, and (4) the use's effect on the market for or value of the copyrighted work.[268] Courts often give most weight to the purpose and character of the use factor—i.e., whether the use is "transformative" or not—as well as the use's effect on the market for or value of the copyrighted work.[269] But no one factor is dispositive.[270] The following sections discuss how the collaborative, interconnected nature of much modern software innovation might be better reflected when taking these four factors into account.

1. Factor One—Whether the Use Is "Transformative"

As mentioned, courts are much more likely to find fair use if the user of the copyrighted work makes a "transformative" use of the copyrighted material. While it is difficult *ex ante* to know whether a particular use is transformative, the inquiry generally focuses on whether "the new work merely 'supersede[s] the objects' of the original creation" or whether and to what extent it alters the original "with new expression, meaning, or message."[271] Hence, if the subsequent user of the copyrighted material adds no new expression or meaning to the original, and fails to put the copyrighted material to novel uses,

---

[267] 750 F.3d 1339, 1376–77 (Fed. Cir. 2014).

[268] 17 U.S.C. § 107 (2012).

[269] *See* Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 579 (1994) ("[T]he more transformative the new work, the less will be the significance of other factors . . . ."); 4 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.05[A][4] (stating that the fourth factor often "emerges as the most important, and indeed, central" factor in fair use cases) (citations omitted)); Joel L. Hecker, *The Wave of the Future or Blatant Copyright Infringement?*, N.Y. ST. B. J. 44, 45 (2007) (indicating that courts have traditionally given the most weight in a fair use analysis to the first and fourth factors).

[270] *Campbell*, 510 U.S. at 577–79; Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 455 n.40 (1984) (both indicating that no one factor is dispositive).

[271] *Campbell*, 510 U.S. at 577–78.

that user is less likely to have engaged in a transformative use of the copyrighted material. And a successful fair use defense becomes, accordingly, less likely. Conversely, if the subsequent user puts the copyrighted materials to uses that go beyond the original aims of those materials—and thereby adds new meaning to them—that user is more likely to have engaged in a transformative use and thus be eligible for a successful fair use defense.

By way of example, courts have often found parodies of copyrighted works to be transformative because a parody uses a copyrighted work to comment on it critically.[272] Parodic uses, therefore, add new meaning or expression to the original work and do not simply mimic it.[273] Conversely, copying and archiving magazine articles for one's later research purposes may not be transformative because the copies serve the same purposes as the original articles.[274]

In the software context, some courts have found transformative use when parties reverse engineer their competitors' software in order to gain access to functional elements of the software for use in creating competing products.[275] These competing products were considered transformative because, although commercially competitive, they were different from, and perhaps improvements upon, their competitors' products.[276] Furthermore, the competitive products did not include copyrightable material from the copyright holder, but functional elements of the software instead.[277] These earlier cases thus point to some fair use case law finding that making intermediate copies of copyrighted works in order to create competitive goods that can interoperate with the competitor's products involves a transformative use.

But the interoperability and collaboration needs of the modern software industry go beyond what these cases may support. Indeed, these earlier cases often have a similarly narrow fact pattern: a party copies its competitor's software in order to gain access to functional, non-copyrighted parts of the software that are necessary to create products that are compatible with those of the competitor's. Furthermore, these cases also seem to depend on a finding that part of what was copied and used in the competitive product was not subject to copyright at all. But what happens if the copying is done simply in

---

272 *Id.* at 579.
273 *Id.*
274 Am. Geophysical Union v. Texaco Inc., 60 F.3d 913, 924–25 (2d Cir. 1995).
275 Sony Comput. Entm't., Inc. v. Connectix Corp., 203 F.3d 596, 602–03 (9th Cir. 2000).
276 *Id.* at 602, 606.
277 *Id.* at 602–03.

order to promote interoperability and collaboration more generally? And what should the outcome be if the copied software is more extensive than a few lines of functional code?

The *Oracle v. Google* case implicates some of these very questions. As discussed, Google copied the Java APIs in significant part because programmers were accustomed to programming software in the Java programming language and using the associated Java APIs. Hence, because the Java APIs had become an industry standard, Google wanted to include that standard in Android so that developers could more readily collaborate with the company in creating compatible software products.[278] Google also copied the elaborate taxonomy of thirty-seven Java APIs in their entirety, rather than merely a few snippets of software code.[279] While these APIs were still functional in important respects, their creation and organization into an extensive taxonomy makes it easier to distinguish their uses from those implicated in earlier software fair use cases.

It is thus unclear, at least based on existing precedent, whether these broader types of collaborative, interconnected uses are transformative under the traditional fair use inquiry. Of course, as others have argued, it may be the case that such uses should be permitted for a different reason altogether: the functional software elements copied should not be subject to copyright protection at all.[280] But as mentioned, it may be more difficult to sustain this type of argument when a party, rather than copying some minimal amount of functional code so that its own products work on a competitor's platform, instead copies an elaborate software taxonomy from another party's product in order to create a product that in some sense displaces that other party's technologies.[281] Indeed, as the Federal Circuit noted in *Oracle v. Google*, such elaborate software taxonomies certainly entail significant amounts of creativity, and clearly enough under copyright's low threshold to qualify for copyright protection.[282]

---

[278] The Federal Circuit, in fact, held this against Google, calling Google's interoperability arguments "confusing" because Google "designed Android so that it would *not* be compatible with the Java platform." Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014). The broader interoperability rationales of Google, therefore, did not hold much sway with the Federal Circuit. *Id.*

[279] *Id.* at 1347.

[280] *See, e.g.*, Samuelson, *supra* note 192, at 702.

[281] *Oracle*, 750 F.3d at 1376–77.

[282] *Id.* at 1364–71.

Hence, if these types of software technologies are subject to copyright and existing fair use cases provide insufficient support for finding that their reuse is protected, then fair use needs a modern reboot so that fair use better facilitates modern software innovation. This Article proposes that uses for purposes of interoperability, broadly defined, should generally weigh in favor of finding a use to be transformative. Interoperability broadly defined naturally includes using another party's software in order to make one's products compatible with the copyright holder's goods and services, as many of the older cases find. But permissible uses would also include reusing another party's software technologies in order to increase compatibility with the broader universe of software technologies, similar to what Google did with the Java APIs.

This broader acceptance of interoperability in the transformative use analysis should not be confined, however, to software interfaces. As discussed throughout, object-oriented programming is an important foundation of collaborative software innovation. But this building-block approach to software development also means that some copyright holder of a software object within a particular software stack could become an obstacle to the entire stack's use. Hence, parties using the stack should have greater leeway to use software objects within it to the extent that these parties add new, transformative expression to the original object, and also to the extent that the software object has become an industry standard so that its absence would frustrate collaborative innovation more generally.

This broader acceptance of interoperability in the transformative use inquiry aligns with the inquiry's goal of ensuring that subsequent uses of the copyrighted work supersede the purposes of the original creation or add new expression, meaning, or message to it. For instance, reuse of software technologies such as software interfaces or objects in order to promote compatibility more generally will often result in the use of these software technologies in completely new contexts, such as enabling otherwise distinctive software services to exchange data in an ever-expanding Internet of Things economy.

Another example is Google's use of the Java APIs. As discussed, the original purpose behind the Java APIs was to allow for compatibility with specific Sun-created Java technologies.[283] Google's reuse of the Java APIs in order to facilitate greater compatibility and collaboration outside of strictly

[283] *Id.* at 1348.

Sun/Oracle products thus represents a different purpose than that of the original creation, and arguably one with greater societal potential. Indeed, according to some accounts, Android has completely transformed the mobile computing industry and powered innovation in the smartphone market since its introduction in 2008.[284]

Of course, if a party copies third-party software in the name of interoperability and collaboration, but in reality simply invokes these principles in order to reproduce the copyrighted holder's own software product, then such uses would clearly not be transformative. For instance, if Google had copied the Java APIs as well as the "implementing code" for carrying out the functions that the APIs specified, a finding of transformative use would be unwarranted. In such a case, Google would have simply copied Oracle's product and sought, with perhaps some minimal changes, to displace it on the market.

But in Android's case, Google created its own implementing code, virtual machine, and Java-based software platform and added many of its own APIs and software objects from third parties.[285] As mentioned, Android incorporated the Java APIs because they represented a widely used programming nomenclature. But there is little if any credence to a claim that Android is merely a copy of some Oracle product. Indeed, Oracle and its predecessor, Sun, had largely failed to create with the Java APIs what Google built with Android.[286] It is thus difficult to argue that Google's use of the Java APIs in Android served the same purpose as Oracle's use of the APIs with its technologies, or that Android's use thereof failed to add new meaning or expression.

Hence, where parties such as Google use another party's software technologies to create a vastly different software program that is compatible

---

[284] *See* Glenn Chapman, *Analysts Say Google Is 'Just Trying Harder' Than Apple, and Android Innovation Is Racing Ahead*, BUS. INSIDER (Nov. 18, 2012, 5:31 PM), http://www.businessinsider.com/android-innovation-is-faster-than-apple-2012-11 (suggesting that Android innovation has outpaced the competition since its introduction in 2008); Anton Wahlman, *Apple Desperately Copies Google's 2008 Features but Passes on Innovation*, THE STREET (June 6, 2014, 5:25 PM), http://www.thestreet.com/story/12730613/1/apple-desperately-copies-googles-2008-features-but-passes-on-innovation.html (suggesting that, in 2014, Apple's most recent improvements to its iPhones simply mimic innovations that Google introduced with Android at its inception).

[285] *See generally* Ron Amadeo, *The History of Android*, ARSTECHNICA (June 15, 2014, 9:00 PM), http://arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/.

[286] Larry Dignan, *Google: Oracle, Sun Blew It on a Java Smartphone*, CNET (Apr. 18, 2012, 5:46 AM), http://www.cnet.com/news/google-oracle-sun-blew-it-on-a-java-smartphone/.

with developers' software applications and development needs more generally, then such uses appear to be transformative in important respects because they "supersede the objects of the original work" and add "new expression, meaning, or message" to the underlying work.[287] Indeed, if the touchstone of the transformative use inquiry is societal benefit resulting from the new uses, then allowing uses that promote interoperability and collaboration broadly defined has much in its favor.

Of course, expanding what counts as a transformative use always runs the risk of colliding with a copyright holder's right to prepare derivative works, one of a copyright holder's exclusive rights.[288] But this seems less concerning when reusing modular software objects. They are by definition self-contained, and the newcomer, in connecting their new object with these preexisting objects, does not "recast, transform, or adapt" the underlying work in a way that the Copyright Act would seem to cover.[289] Similarly, software interfaces simply allow modular products to interface; users of those interfaces thus typically do not modify them, but instead use them as they are to enable distinct software programs to interoperate in new, creative ways. Hence, while a broad reading of "derivative works" could capture these types of uses, an arguably better reading is that these types of uses are not derivative of the underlying works in any vital respect.

2. *Factor Two—Nature of the Copyrighted Work*

The second factor of a fair use analysis concerns "the nature of the copyrighted work."[290] This factor recognizes "that some works are closer to the core of intended copyright protection than others," with the consequence that fair use is more difficult to establish when highly creative works are copied.[291] Examples of highly creative works may be fictional short stories[292] or movies.[293] Furthermore, if a work has not yet been disseminated publicly, then

---

[287] Campbell v. Acuff-Rose Music, 510 U.S. 569, 576–79 (1994).

[288] 17 U.S.C. § 106(2) (2012).

[289] The definition of "derivative work" under the Copyright Act is somewhat amorphous and broad. *See* 17 U.S.C. § 101 (2012). But the examples and criteria provided seem to indicate that the work must somehow be modified, which in the case of modular software products is not the case. *See id.* Indeed, that is one of the primary virtues of object-oriented programming—one can reuse preexisting software objects with new ones without having to modify the preexisting ones.

[290] 17 U.S.C. § 107 (2012).

[291] *Campbell*, 510 U.S. at 586.

[292] Stewart v. Abend, 495 U.S. 207, 237–38 (1990).

[293] Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 445 n.40 (1984).

copying that work is less likely to be found fair because of its unpublished nature.[294] Conversely, if the copied work is mostly factual in nature—for instance, a biography or news broadcast—or has been published, then these factors are more likely to weigh in favor of fair use under the second factor.[295]

When assessing this factor in the software context, some courts have noted that software poses unique challenges because functional considerations often influence a software work's design.[296] And works that have "strong functional elements" are similar to factual works in that they are not entitled to as much copyright protection.[297] Hence, in the context of a software copyright dispute, this second factor will often weigh in favor of finding fair use because of software's functional characteristics.

But the collaborative nature of much modern software innovation, as well as growing interoperability needs within the software industry, have not received explicit attention under this second factor in assessing fair use in the software context. They should for at least two reasons. First, the U.S. Supreme Court has made clear that when works are unpublished, a fair use defense to copying that work is less likely to succeed.[298] A corollary to that premise is that published works should receive greater fair use consideration.[299] In the case of collaboratively built software, this condition is clearly met. Indeed, one of the key points of collaborative models of software innovation is to make the software widely available to the public.[300]

Second and relatedly, these types of software works have been made available with a clear intent to encourage widespread use of the software. In other words, not only have such works been published, they have been published in a manner that abandons typical limitations on use and instead grants users broad freedoms in hope of encouraging ongoing use of the software. As discussed, copyright holders have made available significant numbers of important software works in this manner.[301] Indeed, Sun made many of its Java technologies available under liberal terms as far back as

---

294 Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 563–64 (1985).

295 *Id.*; *cf.* Jake Linford, *A Second Look at the Right of First Publication*, 58 J. COPYRIGHT SOC'Y U.S.A. 585 (2011) (agreeing that a work's publication pushes in favor of fair use but questioning the logic of this result).

296 Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1993).

297 *Id.*

298 *Harper & Row*, 471 U.S. at 563–64.

299 *But see* Linford, *supra* note 295.

300 *See supra* Part II.A.

301 *Id.*

2006.[302] And in the face of Google's adoption of the Java APIs, Sun took no adverse action against Google, even seeming to encourage the use.[303] It was only later, when Oracle acquired Sun and its assets, that Oracle brought a copyright lawsuit against Google.

Hence, when liberally released software is copied but then later becomes subject to a copyright dispute, as in *Oracle v. Google*, courts should take into account as part of the fair use inquiry the permissive manner in which copyright holders have made the works publicly available, either expressly through licensing terms or tacitly through acquiescence to use. And typically, these considerations should weigh in favor of fair use under the "nature of the copyrighted work" factor.

This is not to argue that liberally released copyrighted software should become, effectively, public domain material not subject to copyright at all. Indeed, liberally licensed works often include conditions of use that are important to the copyright holder.[304] But if a party releases copyrighted software works to the public under liberal terms (or perhaps no express terms at all), then arguably the nature of the work changes given the circumstances of its release. Hence, the work, while still subject to copyright, should also become subject to the ongoing collaboration and interoperability needs of users that may have adopted the technologies precisely because of such terms.[305]

Taking into account the functional realties of software can help play a role in policing how broad these ongoing rights should be. As mentioned, the functional nature of software already plays a role in assessing fair use in software disputes under the "nature of the work" factor. And it would continue to do so under this Article's rebooted conception of fair use. For instance, the more functional the software elements copied, the more likely that the "nature of the work" factor should weigh in favor of finding fair use. And this favorable fair use outcome should be even more likely when the functional

---

302 Martens, *supra* note 231.

303 Farber, *supra* note 31.

304 Moglen, *supra* note 107, at 21.

305 In certain respects, this argument may be likened to a promissory estoppel argument under contract law, where parties, despite the absence of a bargained-for contract, may still have some quasi-contractual rights vis-à-vis the promisor because of reasonable reliance on the promise and considerations of equity. David G. Epstein, Melinda Arbuckle & Kelly Flanagan, *Contract Law's Two "P.E.'s": Promissory Estoppel and the Parol Evidence Rule*, 62 BAYLOR L. REV. 397, 404–07 (2010). Similarly, in the software context, where a party releases software to the public with express or tacit hopes of encouraging use thereof, the fair use inquiry under the "nature of the work" factor should take this into account in determining whether users should have ongoing rights with respect to the software.

software elements have been liberally released to the public, as discussed above.

In sum, liberally released software elements that enable interoperability and collaboration, such as APIs and certain software objects, are one good candidate for a fair use finding based on the "nature of the copyrighted work" factor within the fair use inquiry. At least two reasons support this conclusion. First, these types of software elements are often strongly functional; they are necessary to use in order to ensure compatibility with other software that relies on the same or similar technologies. And second, because the copyright holders in question initially released the functional software elements under liberal terms (or otherwise tacitly encouraged use thereof), it seems inequitable to then allow these same copyright holders to switch their stance once users have taken the proffered bait.

3. Factor Three—Amount of the Copyrighted Work Used

Factor three in the fair use inquiry takes into account the "amount and substantiality of the portion used in relation to the copyrighted work as a whole."[306] In general, the more of a copyrighted work that another party copies, the more difficult it becomes to sustain a fair use defense.[307] But even copying relatively small amounts can weigh against fair use if the copier takes "the heart" of the copyrighted work.[308] On the other hand, some courts have still found fair use where a party copies the entire copyrighted work.[309] Such outcomes are still possible when, despite the wholesale copying, fair use's other factors strongly militate in favor of fair use (e.g., because the use is considered highly transformative).[310]

The collaborative, interconnected nature of much modern software innovation should become more relevant to assessing this factor of the fair use inquiry as well. For instance, copying software interfaces for compatibility or collaboration purposes constitutes a limited amount of copying of the underlying copyrighted work; interfaces divorced from the software that

---

306 17 U.S.C. § 107 (2012).

307 *See* NIMMER & NIMMER, *supra* note 269, at § 13.05[A][3].

308 Harper & Row, Publishers v. Nation Enters., 471 U.S. 539, 564–65 (1985) (finding that copying even small portions of an unpublished book weighed against fair use because the second user copied the "heart of the book" (quoting Harper & Row, Publishers v. Nation Enters., 557 F. Supp. 1067, 1072 (S.D.N.Y. 1983))).

309 Sega Enters. v. Accolade, Inc., 977 F.2d 1510, 1526–27 (9th Cir. 1992) (citing Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 449–50 (1984)).

310 *Id.* at 1527.

actually implements them are only a small part of the larger software work. Furthermore, interfaces are not the "heart" of the work, either. Instead, they are simply a taxonomy of functions that many developers are either accustomed to using or need to use so that their own software products are compatible with technologies that employ the same interfaces. The heart of the software work remains the software that performs the specified functions, both in the original product and those products meant to interoperate with it through such interfaces.

Of course, courts may frame the issue differently by viewing interfaces as distinctly copyrightable works. For instance, the Federal Circuit in *Oracle v. Google* appeared to view the APIs as a separately copyrightable work distinct from the software that actually implements the computing functions that the APIs specify.[311] Under this view, copying the interfaces would thus constitute copying the entire work. Furthermore, this characterization of the work may also mean that the "heart" of the work was copied, since the copier uses the interfaces for similar purposes for which they were originally designed (i.e., as interfaces). Hence, if this characterization were to be accepted, it would likely militate against a finding of fair use.

But viewing software interfaces in this way seems misguided for several reasons. For instance, the Java APIs, and interfaces in general, serve no purpose without software that implements the functions they specify; the purpose behind interfaces is simply to identify commands that other software performs.[312] Hence, if no actual software implementing the interfaces is present, the interfaces on their own have no value or purpose. Viewing the interfaces divorced from the underlying software works, therefore, is in some sense nonsensical because the interfaces have no independent practical reality, even if they are identifiable in software source code files.

Of course, it is certainly possible to separate a single copyrightable work into multiple, distinct copyright interests. A collective work such as a newspaper, for instance, includes distinct copyrights in each article, photo, and other creative material present in the newspaper issue, as well as a copyright in the overall newspaper issue.[313] Well-delineated characters, such as Superman or Rocky, can also be separately copyrightable from the works in which they

---

311 *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1367–68 (Fed. Cir. 2014).

312 Proffitt, *supra* note 222.

313 U.S. COPYRIGHT OFFICE, FL-104 CONTRIBUTIONS TO A COLLECTIVE WORK (2016), http://www.copyright.gov/fls/fl104.pdf.

appear.[314] Furthermore, copyright merely requires that a fixed work entail "independent creation" and "a modicum of creativity"; it includes no specific requirement that a work be useful, independently or otherwise.[315]

But software is difficult to analogize to other types of copyrightable works because of its utilitarian nature. For instance, with a newspaper, each separate article or photo still has some purpose when separated from the overall newspaper issue. An article still serves the purpose of providing whatever message it conveys. And an accompanying photo still conveys some information to the beholder of it, even if the article with which it was originally paired adds context to that message.

Software interfaces, conversely, are strictly functional in carrying out the specified functions and facilitating communication between software products—if those software products are absent, the interfaces themselves do nothing and convey no information to anyone that happens to observe them in software source code. Hence, whatever creativity interfaces entail only becomes present and relevant when they are paired with the software that implements them.

Distinguishing copyrighted characters from software interfaces drives home this point. For instance, characters in works only earn copyright protection when the characters have been sufficiently delineated in the works in which they appear.[316] Or as one court put it, delineation of the character must be such that the character constitutes "the story being told."[317] While characters can thus in some sense become independently copyrightable from the underlying works, in reality their copyright depends on how the author has developed the character in the underlying works. Indeed, even when well-delineated characters such as Rocky appear in a sequel, they carry with them into the sequel the characteristics delineated in the previous films.[318] Without reference to those underlying works, therefore, it is nonsensical to discuss copyright in those characters.

---

[314] *See* Zahr K. Said, *Fixing Copyright in Characters: Literary Perspectives on a Legal Problem*, 35 CARDOZO L. REV. 769, 772–73 (2013).

[315] Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 346 (1991) (citing *In re* Trade-Mark Cases, 100 U.S. 82, 94 (1879)).

[316] *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

[317] Warner Bros. Pictures v. Columbia Broad. Sys., 216 F.2d 945, 950 (9th Cir. 1954).

[318] *See* Said, *supra* note 314, at 815 n.235.

Software interfaces are different. When they appear in their version of a "sequel"—a new software work such as Android, for instance—the implementing software from the prior work that initially gave them an identity is no longer present or even relevant. Instead, the new software that Google or some other third party writes to implement the interfaces gives those interfaces their new identity and purpose. Software interfaces may thus be likened to copyrighted characters in that their copyright status depends on the software that implements them. Without that implementing software, software interfaces have no identity or purpose. But that is where the similarities with copyrighted characters stop because software interfaces do not rely on earlier software implementations for their identity in new contexts, such as when Google used the Java APIs in Android. Instead, the new software context—Android—provided those software interfaces a new reality. And that new reality as a whole is what, arguably, should be subject to copyright.

Hence, the third factor under the fair use inquiry—the amount and substantiality of the copyrighted work copied—should weigh in favor of fair use when parties copy software strictly for interoperability and collaboration purposes. And this conclusion is justified for at least two reasons. First, the amount copied—typically software interfaces—is minimal when compared to the software code necessary to implement the interfaces. Courts may reach different conclusions if they treat software interfaces as distinctly copyrightable works. But as discussed above, software interfaces should be viewed together with the implementing software as a single work, since breaking the work into component parts renders the separate works inoperable. Second, while interfaces are important functionally, copying them for compatibility and collaboration purposes does not result in the "heart" of the work being used. The heart of the software work consists of actual implementation of the computing functions that the interfaces specify.

When software developers copy software objects wholesale, however, much of the above analysis changes. For instance, it is harder to argue that the object is simply a small piece of a larger software stack, since the object is modular and self-contained. Indeed, the copier, in replicating the object wholesale, has almost certainly copied the "heart" of the modular object; it is hard to imagine what the software object's heart would otherwise be. Such copying may thus push against fair use under this factor, though if the copying involves a transformative use with minimal effects on the market, this factor may carry less weight in the overall fair use decision.

4. Factor Four—Effect on the Market

The fair use inquiry's fourth factor—the use's effect on the market for or value of the copyrighted work—is traditionally one of the most important factors within the fair use probe.[319] This factor requires courts to assess not only the market harm resulting from the alleged infringer's actions, but also to consider the impact on potential markets for the copyrighted work if uses similar to that of the alleged infringer were to become widespread.[320] If the fair use proponent uses the copyrighted material for commercial purposes, this may weigh against a finding of fair use, particularly if the use involves mere duplication of the copyrighted work for commercial purposes.[321] But where a work is transformative under fair use's first factor, the commercial nature of a use is not itself dispositive.[322]

Hence, if a party simply copies and distributes verbatim another party's copyrighted work, then the fourth factor—and the fair use question more generally—is often easy to resolve against the purported fair user.[323] More difficult to resolve are cases where courts must assess not only existing sales and markets, but potential markets as well.[324] In these scenarios, courts sometimes engage in a form of circular reasoning by simply concluding that a user's failure to license a copyrighted work from the copyright holder cuts against fair use because that failure harms the copyright holder's market for the work.[325] In other instances, courts decide in favor of fair use, without much detailed analysis, simply because use of the copyrighted work involves criticism of it.[326] Because both categories of decisions often lack analytical depth, some scholars view this fourth factor as one of the more problematic ones within the fair use inquiry.[327]

---

[319] 17 U.S.C. § 107(4) (2012). For the importance of this factor, see NIMMER & NIMMER, *supra* note 269, at § 13.05[A][4] (stating that the fourth factor often "emerges as the most important, and indeed, central" factor in fair use cases (footnotes omitted)).

[320] Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 590 (1994) (quoting NIMMER & NIMMER, *supra* note 269, at § 13.05[A][4]).

[321] *Id.* at 590–91 (citing Acuff-Rose Music, Inc. v. Campbell, 972 F.2d 1429, 1438 (6th Cir. 1992)).

[322] *Id.* at 591.

[323] *See, e.g.*, United States v. Slater, 348 F.3d 666, 669 (7th Cir. 2003) (rejecting the defendant's plea for fair use in a similar situation).

[324] Carroll, *supra* note 43, 1104–05 (describing this difficulty).

[325] Jeanne C. Fromer, *Market Effects Bearing on Fair Use*, 90 WASH. L. REV. 615, 616 (2015).

[326] *Id.*

[327] WILLIAM F. PATRY, PATRY ON FAIR USE § 6:1 (2016) (indicating that the fourth factor is poorly understood and, as a result, often misapplied).

Several scholars have recently argued that one way to help address such problems is for courts to explicitly take into account market benefits as well as market harms.[328] For instance, courts often focus on the use's potential harm to the market for the copyrighted work.[329] But in reality, market benefits arising from the use are also relevant because, if present, they may more definitively establish that no market harm exists. In the recent Google Books litigation, for instance, one way to demonstrate a lack of market harm is to show that the use—digitizing millions of books and then making snippets thereof available through search queries—actually boosts book sales.[330]

While market benefits arising from the use may influence courts' reasoning, they are not typically relied on in courts' fourth factor analyses.[331] But as Jeanne Fromer notes, the Supreme Court has provided some endorsement for an approach that explicitly considers market benefits, and she argues that society would be well served if courts more fully embraced it.[332] Fromer also points to Supreme Court case law that implies two ways to separate relevant market effects from irrelevant ones: courts should exclude from consideration those effects that are "empirically unlikely," as well as market effects that "are unrelated to the protectable aspects of the copyrighted work, such as its ideas or the societal value attributed to the work."[333]

This approach to applying the fourth fair use factor, if consistently adopted, would help better take into account the collaborative, interconnected nature of much modern software innovation. For instance, if Party A borrows interfaces and related elements from Party B's software in order to make its software compatible with Party B's products, both market benefits and harms are relevant to the fair use inquiry. On the one hand, Party B may miss out on a licensing opportunity with Party A. That missed opportunity may be viewed as a market harm if Party B regularly licenses these software components to third parties, though it is also true such markets may exist primarily because parties

---

328 *See, e.g.*, David Fagundes, *Market Harm, Market Help, and Fair Use*, 17 STAN. TECH. L. REV. 359 (2014); Fromer, *supra* note 325.

329 *See, e.g.*, Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 590 (1994) (quoting NIMMER & NIMMER, *supra* note 269, at § 13.05[A][4]).

330 Joseph Ax, *Google's Book-Scanning Project Legal, Says U.S. Appeals Court*, REUTERS (Oct. 16, 2015, 2:09 PM), http://www.reuters.com/article/us-google-books-idUSKCN0SA1S020151016.

331 *Cf.* Fromer, *supra* note 325, at 617 (noting that "twenty years after *Campbell*, some courts have begun to recognize that market benefits ought to count in favor of finding that a defendant's use is fair").

332 *See id.* at 629.

333 *Id.* at 618, 641–49.

are often risk-averse and simply obtain licenses to third-party products that the law may not actually require.[334]

But Party B's software may also benefit from compatibility with Party A's product. Indeed, this benefit may be particularly so in a world where consumers increasingly expect heterogeneous goods and services to be able to exchange data, such as with the Internet of Things.[335] Hence, in a collaborative, interconnected software world, these types of market benefits should also be taken into account in resolving how use of the copyrighted work affects the work's market.

Situations where parties use software from another party in order to increase compatibility more generally require a different analysis under this factor. Google's use of the Java APIs may be characterized as this form of borrowing. As discussed, Google incorporated the Java APIs into Android in order to increase the compatibility of Android with the development practices and products of software developers more generally.[336] But its use of the Java APIs did not make it compatible with Oracle's Java software technologies.[337] Hence, possible market harms and benefits in such scenarios differ from those where a party uses software elements from a third party simply to achieve compatibility with that third party's products.

In these types of scenarios, on first glance market harm may seem relatively straightforward. For instance, in *Oracle v. Google*, Oracle contended that Google's use of its technologies in Android undermined Oracle's ability to license these same technologies to others.[338] Hence, while Oracle "never successfully developed its own smartphone platform using Java technology," Android's release made it impossible for Oracle to license others to do so, or to eventually do so itself.[339] Furthermore, there is undisputed evidence that Sun and Google engaged in lengthy licensing discussions with regards to the APIs, which negotiations ultimately broke down, thereby eliminating a licensing opportunity for Sun.[340] Consequently, market harm to Oracle and other

---

334 *See generally* Gibson, *supra* note 211 (arguing that risk aversion may generally lead to an expansion of intellectual property rights, or at least how parties and courts perceive the scope of intellectual property rights in determining whether permission to use third-party materials is necessary).

335 *See supra* note 139 and accompanying text.

336 *See supra* notes 224–227 and accompanying text.

337 *Id.*

338 *See* 750 F.3d 1339, 1377 (Fed. Cir. 2014).

339 *Id.* (quoting Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012)).

340 Farber, *supra* note 31.

similarly situated parties may result because copying those parties' technologies eliminates market opportunities that they otherwise would have had.

But market harm in such cases is not so straightforward. Examining certain aspects of the *Oracle v. Google* case helps exemplify why. First, while Oracle may have lost out on a licensing opportunity with Google,[341] there is as yet no clear evidence that other parties were seriously interested in licensing the Java technologies for building a smartphone platform, or that their interest waned once Android was released.[342] Instead, the available evidence suggests that Oracle's Java solution for smartphones simply did not work very well, which seems to be the more likely reason why parties may not have been as eager as Sun (and later Oracle) hoped to license the technologies for building a smartphone platform.[343]

Furthermore, even with respect to the Google licensing opportunity, some evidence suggests that Sun did not consider the APIs proprietary at the time, instead hoping to license the Java trademark to Google so that Android-based devices were Java-branded.[344] Indeed, Sun may have even been willing to pay Google to adopt the Java platform.[345] Hence, the alleged negative market effects of Google's use of the Java APIs seem "empirically unlikely," or at least highly questionable.

Second, as discussed, Google used functional elements of the Java technologies to make its product more appealing to software developers. While the Federal Circuit found that the Java APIs included sufficient creativity for them to be subject to copyright,[346] the Java APIs' functional nature nonetheless

---

[341] Brandon Bailey, *Larry Page Evasive with Oracle's Lawyer, but Admits Google Never Obtained Java License*, MERCURY NEWS (Apr. 18, 2012, 3:55 AM), http://www.mercurynews.com/ci_20424638/google-oracle-trial-larry-page-admits-android-java-licence (detailing how Sun and Google engaged in licensing negotiations with respect to the Java technologies, but ultimately failed to reach a deal).

[342] *Cf.* Larry Dignan, *Google: Oracle, Sun Failed at Java Smartphone Now Stop Whining*, ZDNet (Apr. 18, 2012, 5:22 AM), http://www.zdnet.com/blog/btl/google-oracle-sun-failed-at-java-smartphone-now-stop-whining/74561 (noting that "Sun failed to popularize Java based smartphones").

[343] *Id.* (highlighting Google's evidence that Sun/Oracle had tried but failed multiple times to develop a Java-based software platform for smartphones); Andrew Orlowski, *Java Won the Smartphone Wars (and Nobody Noticed)*, THE REGISTER (July 19, 2012, 9:02 AM), http://www.theregister.co.uk/2012/07/19/java_java_everywhere/ (describing how Java-based phones largely failed because of sluggish performance, but how certain Java technologies have proved vital to much of the smartphone market because developers are accustomed to using Java in developing software apps).

[344] Farber, *supra* note 31.

[345] *Id.*

[346] *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1363 (Fed. Cir. 2014).

should mean that they enjoy less expansive copyright protection than they otherwise would. Hence, whatever market harms Oracle suffered because of Google's use of the Java APIs stem from aspects of Oracle's software that are less deserving of expansive copyright protection.

Third and importantly, taking into account the other fair use factors in connection with the fourth factor may make purported market harms even more dubious. For instance, as discussed, Google only copied thirty-seven APIs out of hundreds and wrote its own implementing software, virtual machine, and other APIs for Android.[347] Hence, Google did not merely duplicate Oracle's smartphone solution in the marketplace; Oracle did not possess one to supersede, at least that third parties were eager to adopt. Instead, Google arguably put some of Oracle's Java technologies to a transformative use by utilizing them to create something that the market at the time lacked, and which has yielded significant societal benefits.[348]

Of course, another view is that Google merely duplicated Oracle's entire copyrighted work—the Java APIs—and that duplication superseded the purposes of the original work and simply "serves as a market replacement" for the APIs.[349] But as discussed above under the third factor of the fair use inquiry, separating software interfaces from the underlying software product is in important respects illogical since the interfaces have no independent practical reality without the software that implements them.

Furthermore, Android is not a substitute in the market for the Java APIs. If anything, Android is a substitute for something neither Sun nor Oracle was able to achieve: a successful smartphone software platform. Of course, Google's use of the APIs, if ultimately condoned, may mean that Oracle will face significant difficulty persuading another party to license the APIs. Hence, if Google's use were to become widespread, it may mean that potential markets for the APIs disappear.[350] But potential markets that are "empirically unlikely" arguably should not carry significant weight,[351] and there is no evidence that Oracle has had, currently has, or is poised to have a licensing market with respect to the APIs separate from the software that implements

---

[347] Oracle Am. Inc. v. Google Inc., 872 F. Supp. 2d 974, 978–79 (N.D. Cal. 2012), *aff'd in part, rev'd in part*, 750 F.3d 1339 (Fed. Cir. 2014).

[348] *See supra* note 284 and accompanying text.

[349] Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 590–91 (1994) (citing Sony Corp. v. Universal City Studios, 464 U.S. 417, 451 (1984).

[350] *See id.*

[351] Fromer, *supra* note 325, at 642–45.

them.[352] Indeed, this reality is not terribly surprising since APIs are not truly separate works from the underlying software and only have meaning—and value—when accompanied by software that performs the computing functions that the APIs specify, as discussed above.

Fourth and relatedly, Google's use of the APIs may actually provide Oracle with some market benefits. For instance, by further cementing the Java APIs as industry standards, Google's use may improve the market for Java technological solutions that Oracle actually possesses. Indeed, if Google had opted for a different API nomenclature in Android, one possible result is that more developers would, over time, wean themselves from using the Java APIs. And if that happened, Oracle may have greater difficulty in attracting customers to the Java-based solutions that it does offer.

In sum, fair use's fourth factor should better take into account the collaborative, interconnected nature of much modern software development. Courts can do so by assessing both market harms and benefits of the use, as well as ruling out "empirically unlikely" potential markets and granting less copyright protection when uses pertain to functional aspects of the software work. Furthermore, this Article's reassessment of the first three fair use factors should play into how courts assess this final fair use consideration as well. Doing so may lead to different outcomes in cases where software innovation's collaborative, interconnected realities are highly relevant, but often forgotten.

IV. BEYOND SOFTWARE'S COPYRIGHT PROBLEMS

The software industry is clearly not the only context in which collaborative, interconnected creativity is increasingly relevant. As others have noted, the advent of the Internet and digital technologies has enabled collaborative, interconnected creativity in a variety of other spheres as well, including music, photography, video, and literature.[353] This "read-write" culture, where both professional producers and everyday consumers interact to create

---

[352] *See Java SE General FAQs*, ORACLE, http://www.oracle.com/technetwork/articles/javase/faqs-jsp-136696.html (last visited Feb. 5, 2016); *Licensing and Distribution FAQs*, JAVA, https://java.com/en/download/faq/distribution.xml (last visited Feb. 5, 2016) (providing licensing details with respect to Java software implementations in general, which includes the interfaces).

[353] *See generally* LAWRENCE LESSIG, REMIX: MAKING ART AND COMMERCE THRIVE IN THE HYBRID ECONOMY (2008) (noting the growth of interconnectivity and its impact under current intellectual property law).

copyrightable materials, is apparent in today's world of YouTube, Facebook, Instagram, and blogging.[354]

Nonetheless, scholars have generally downplayed concerns about a copyright-induced anticommons.[355] For instance, Michael Heller has suggested that "[c]ompared with patent law, copyright law's tragedy of the anticommons is less costly" because copyright's fair use doctrine allows for many uses that would otherwise require permission.[356] He does note that some underuse of copyrighted resources may still occur, despite fair use, because of the deterrent effect of transaction costs, as well as the fact that copyright holders may still seek rents that exceed the value of the fair use.[357] But Heller's fear of a copyright-induced anticommons remains minimal at best.

Heller may have been less concerned with a copyright anticommons in part because today's collaborative, interconnected economy was not the reality in 1999, when he wrote his article. But as argued throughout this Article, that collaborative, interconnected economy is increasingly a cause of anticommons concerns in the software context, particularly since the fair use inquiry is not specifically adapted to these realities. And it stands to reason that similar collaborative, interconnected creativity in contexts outside of software will have comparable anticommons effects in those contexts.

Other scholars have shown less concern about anticommons problems in a collaborative economy because, they contend, many creators in the read-write culture may not even know that they own a copyright, or they create for reasons unrelated to financial concerns.[358] Yet a variety of recent copyright controversies makes clear that these realities do not make anticommons concerns disappear, but instead may be the source of them, in the software context and elsewhere.

For instance, as described in the Introduction, in 2014 Cindy Lee Garcia asserted copyright against Google in hopes of removing a film from YouTube that included a five-second performance by her.[359] The anti-Islamic film had created outrage in many quarters, and Garcia's brief appearance in the film had

---

354 *See generally id.* at 36–50 (providing an overview of this interaction between consumers and producers).

355 *See, e.g.*, Heller, *supra* note 26, at 1175 n.61; Fagundes & Masur, *supra* note 26, at 718.

356 Heller, *supra* note 26, at 1175 n.61

357 *Id.*

358 *See* Fagundes & Masur, *supra* note 26, at 718–19.

359 *See supra* notes 1–11 and accompanying text.

resulted in death threats against her.[360] Her copyright infringement claim was thus meant to protect her privacy and, ultimately, her life.[361] Her first legal recourse, however, was not to copyright.[362] But once she learned that copyright may be the most effective tool for removing the film from the public sphere, she resorted to it.[363]

Hence, for Garcia, copyright became a tool for safeguarding interests unrelated to the purposes of copyright.[364] And when copyright becomes a tool for safeguarding such interests, it becomes a source of anticommons concerns, particularly in a world where copyrighted works are subject to multiple copyright claims. Thus, the mere fact that contributors to a collaborative work may not initially recognize that they have a copyright interest in the work, or do not contribute to the work for financial reasons, does not mean that those contributors will not eventually resort to copyright. Instead, as other scholars have noted, examples increasingly abound of parties attempting to use copyright to protect interests unrelated to copyright.[365] When they do so, they may contribute to an anticommons. And the increasingly collaborative, interconnected nature of the world promises that such anticommons concerns will only grow as the number of copyrights in a given resource increases.

In sum, copyright law—and society—would be well served with adaptations that better reflect these collaborative, interconnected realities in many creative settings. While this Article has focused on addressing how to do so in the software context, similar fair use tailoring may prove useful in other areas of creativity as well. This Article thus joins a growing body of scholars urging that intellectual property law in general must better align with the changing creative and inventive norms present in today's world in order to promote, rather than inhibit, innovative and creative activities.[366]

---

360 *See supra* notes 1–11 and accompanying text.

361 Garcia v. Google, Inc., 766 F.3d 929, 932, 939 (9th Cir. 2014).

362 Asay, *supra* note 202, at 41.

363 *Id.*

364 For an argument as to why the law should care about copyright being used to protect interests unrelated to the purposes of copyright, see Fromer, *supra* note 202.

365 *Id.* at 556–62 (reviewing many such examples).

366 *See, e.g.*, LESSIG, *supra* note 353, at 253–73 (laying out proposals for adapting copyright law to the read-write culture); Mark A. Lemley, *IP in a World Without Scarcity*, 90 N.Y.U. L. REV. 460 (2015) (discussing how IP law must adapt to new models of production that turn the traditional economic theories behind IP law on their head).

CONCLUSION

To the casual observer, the recent *Oracle v. Google* case may appear to concern a rather mundane, technical question of copyright law. But as this Article has argued, the case actually provides an apt illustration of growing anticommons concerns in the increasingly collaborative, interconnected world of software. These problems, if left unchecked, are likely to only grow unless copyright law makes needed adjustments. And those adjustments must center on better balancing the needs of individual copyright owners with those of the hundreds and thousands of others that increasingly collide with each other in today's collaborative, interconnected software world. This Article has recommended certain changes to copyright law's fair use doctrine as one possible means by which to better achieve that balance.

That same balance may also be needed in other areas of collaborative, interconnected creative effort. Indeed, the economic models behind U.S. copyright law, with their focus on the incentives of individual creators, fail to adequately take into account the incentive structure in more collaborative, interconnected models of creative production. But in order to better "promote the Progress of Science and useful Arts," the constitutional basis for copyright law, they must.[367]

---

[367] U.S. CONST. art. I, § 8, cl. 8.