Mills Legal Clinic
Stanford Law School

**Juelsgaard Intellectual Property and Innovation Clinic**
Mills Legal Clinic
Stanford Law School
Crown Quadrangle
559 Nathan Abbott Way
Stanford, CA 94305-8610
jipic@law.stanford.edu
650.724.1900

*Before the*

**U.S. COPYRIGHT OFFICE**
**LIBRARY OF CONGRESS**

In the Matter of Software-Enabled Consumer Products Study

Docket No. 2015-6

# Comments of Engine Advocacy

Sydney B. Lakin
Wangshu Tai
Brian Weissenberg
*Certified Law Students*
Phil Malone
Jef Pearlman
*Counsel for Petitioners*

## Table of Contents

Engine Advocacy ("Engine") submits this comment in response to the Copyright Office's Notice of Inquiry and Request for Public Comment for its Software-Enabled Consumer Products Study.

Engine supports the growth of technology entrepreneurship through economic research, policy analysis, and advocacy on local and national issues. Engine works with the White House, Congress, federal agencies, state and local governments, and also international advocacy organizations to educate and inform them of the changing face of American high-tech entrepreneurship and issues vital to fostering technological innovation.

Engine may be contacted through the above-identified counsel.

## I. Embedded Software in Everyday Devices Creates Tremendous Opportunities for Innovation and Increased Consumer and Social Value

Not long ago, consumer software was typically found in standalone applications and operating systems that ran primarily on desktop or laptop computers. Limited categories of software-enabled consumer products existed, including early video game consoles, calculators, and microwaves, but these were the exception rather than the rule. Today, however, software is everywhere. It is embedded not just in our phones (which really are far more like laptop computers than like traditional landline phones), but in our refrigerators, light bulbs and other home devices, in our cars and tractors, in our pacemakers and other medical devices, in our kids' dolls and other toys, and in many thousands of other everyday products. As the "Internet of Things" rapidly expands, embedded software and software-enabled consumer products will become truly ubiquitous.

The increasing prevalence of software-enabled products and the rise of the Internet of Things offers great value to consumers, businesses, and other users of the vast array of innovative products in which such software will be found. It also offers tremendous opportunities to innovators, entrepreneurs, startups, and other businesses. Some will produce the software-enabled devices and some will provide the embedded software. But countless others will develop and sell products, software, and services that connect and interoperate with the embedded software and devices to enhance, extend, and enrich the user experience. The promise is a host of new platforms and ecosystems that will unleash waves of innovations. This will result in greater competition among products and developers and vastly increased consumer choice and social value.

As has been the case with the Internet itself, however, realizing the full measure of this promise depends on the continued opportunity for what many have called "permissionless innovation."[1] Permissionless innovation means that any user, innovator, or startup can create new software programs, devices, and services that are able to connect, interoperate, and interact with the software embedded in a device without the need to first seek permission or a license from the developer of the original device. A bedrock of the traditional software industry has been

[1] Henry Chesbrough & Marshall Van Alstyne, *Permissionless Innovation*, 58 Commc'n of the ACM. 24, 24 (2015) ("'Permissionless innovation,' is the freedom to explore new technologies or businesses without seeking prior approval. It has already produced an explosion of goods and services in the IT industry.")

the availability and openness of application programming interfaces ("APIs") or other interfaces and software for interoperation, that permit compatibility and interoperability. The same principle is critical to realizing the full promise of embedded software and the Internet of Things.

Social and consumer value is also enhanced by the ability of consumers, user communities, and researchers to test, tinker with, improve, repair, transfer, and resell their software-enabled products in a manner similar to other products they purchase and own. Moreover, although embedding software to make devices "smart" usually means greater functionality, it can also create additional risks. For example, in 2014, security researchers uncovered a loophole in the infotainment system of a Jeep Cherokee that allowed them to remotely disable its transmission and brakes.[2] But with risk comes opportunity; security researchers should be able to explore and analyze the embedded software to detect (as in the Jeep example or the recent Volkswagen emissions software controversy) flaws, security vulnerabilities, hidden privacy risks, etc. Users should be able to tinker with and repair their own devices if they choose. Again, this opportunity depends on the ability of researchers and users to freely and openly connect with, interrogate, and test the software embedded in a variety of devices.

## II. NOI Question 3: Whether and to what extent innovative services are being enabled and/or frustrated by the application of existing copyright law to software in everyday products.

The application of existing copyright law threatens to significantly frustrate innovation, competition, and consumer expectations for and enjoyment of devices they purchase as embedded software becomes more and more a part of the devices used in everyday life. In particular, treating APIs as copyrightable cripples permissionless innovation and interferes with innovators' ability to create products and services that interoperate with existing products and platforms and/or compete effectively with them. Allowing producers of software-enabled devices to use copyright to deny purchasers of those devices ownership of the embedded software frustrates consumer expectations and prevents users, user communities, researchers, and others from analyzing, improving, repairing, or reselling their devices. The result is a drag on innovation and on the development and success of new products and services, and on consumer choice and welfare.

### *A. Background of Interaction Between Copyright Law & Embedded Software*

The Copyright Act of 1976 recognized that significant technological challenges lay not far ahead. The National Commission on New Technological Uses of Copyrighted Works ("CONTU") sought to understand these challenges and prescribe solutions, and Congress accepted CONTU's recommendation that copyright protection be extended to computer programs in 1980. In 1998, Congress enacted the Digital Millennium Copyright Act (DMCA) in part to address circumvention of technical protection measures while recognizing the need for copying for maintenance, repair, reverse engineering, and other appropriate purposes.

[2] Andy Greenberg & Kim Zetter, *How the Internet of Things Got Hacked*, http://www.wired.com/2015/12/2015-the-year-the-internet-of-things-got-hacked/.

With the advent of embedded software, the licensing regime of the digital space began to clash with the ownership culture of the physical world. *Chamberlain v. Skylink* and *Lexmark v. Static Control Components* foreshadowed many of the challenges facing embedded software today. In *Chamberlain,* Skylink produced a replacement garage door opener remote that could interoperate with Chamberlain's garage door opener. In developing the remote, Skylink had to circumvent the original system through reverse engineering. Chamberlain filed suit, alleging that "*all* such uses of products containing copyrighted software to which a technological measure controlled access are now per se illegal under the DMCA unless the manufacturer provided consumers with *explicit* authorization."[3] Skylink responded that, "in the absence of an explicit restriction, consumers must be free to infer that they have purchased the full range of rights that normally accompany consumer products—including those containing copyrighted embedded software."[4]

The court rejected Chamberlain's theory of authorization, holding that the DMCA does not "divest the public of the property rights that the Copyright Act has long granted to the public" and, therefore, individuals were permitted to "*use* (emphasis added) the copy of . . . copyrighted software embedded in the" garage door openers they purchased.[5] This ruling, however, avoided the key question of license versus ownership; the next company in Chamberlain's position could meet the burden of proving unauthorized access by inserting "a single copyrighted sentence or software fragment to its product, wrap(ping) the copyrighted material in a trivial 'encryption' scheme."[6]

In the same year, the Sixth Circuit faced a similar question in *Lexmark v. Static Control Components.* Lexmark, a printer and toner cartridge manufacturer, sold a printer and cartridge duo; embedded in the printer was its Printer Engine Program (PEP) and in its cartridge, the Toner Loading Program (TLP). In order to start printing, the PEP had to establish communication with TLP and performed several routines. SCC copied Lexmark's TLP and created "a cartridge microchip that bypassed each of these measures" and sold that chip to third-party cartridge refillers.[7] Lexmark sued SCC, claiming that its "bypassing of the authentication sequence circumvented technological measures that protected access to two copyrighted works—the PEP and TLP programs—and thus violated Section 1201."[8] The court rejected SCC's 1201 argument on the narrow ground that PEP code was unencrypted and that TLP was not copyrightable. Again, the court avoided the critical question.

In each of these cases, manufacturers embedded software in their consumer products and sought to use copyright law "offensively" rather than for the intended purpose of the DMCA, "to

[3] *Chamberlain Group, Inc. v. Skylink Techs., Inc.*, 381 F.3d 1178, 1193 (Fed. Cir. 2004).
[4] *Id* at 1187.
[5] *Id* at 1204.
[6] *Id*., at 1201.
[7] Michael A. Carrier, Innovation for the 21st Century: Harnessing the Power of Intellectual Property and Antitrust Law, 185 (2010).
[8] *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 544, 552 (6th Circ. 2004).

prohibit the pirating of copyright-protected works such as movies, music, and computer programs."[9]

In the years after *Chamberlain* and *Lexmark*, courts addressed cases that grappled with ownership status under software licenses. *Vernor v. Autodesk*, in the Ninth Circuit, was "one of a number of cases to consider first sale rights in the software context."[10] The defendant purchased "several used copies of Autodesk, Inc.'s AutoCAD Release 14 Software," accompanied by shrink-wrap licensing agreements, and proceeded to resell the copies on eBay.[11] The court considered whether Autodesk had transferred copyright ownership to Vernor, making his reselling lawful under the first sale doctrine, or whether he was a mere licensee infringing Autodesk's copyright. The court found that Vernor could not "invoke the first sale doctrine" because Autodesk's shrink-wrap licensing agreement was valid and that "Autodesk's direct customers are licensees of their copies of the software rather than owners."[12]

The Second Circuit reached a different conclusion in *Krause v. Titleserv*, where it held that "formal title in a program is not an absolute prerequisite for qualifying for" 17 U.S.C. § 117(a)'s safe harbor.[13] Courts should, instead, "inquire into whether the party exercises sufficient incidents of ownership over a copy of the program to be sensibly considered the owner of the copy for purposes of § 117(a)."[14] This decision broadened the meaning of copyright ownership finding that "the absence of formal title may be outweighed by evidence that the possessor of the copy enjoys sufficiently broad rights over it to be sensibly considered its owner."[15]

In recent years, there has been a significant trend toward vendors distributing software-enabled products governed by contracts that state that the software inside is licensed to the purchaser of the product rather than owned by them. In one very recent example, John Deere and General Motors advanced a view of embedded software that challenges the very idea of owning a tractor or a car: the purchaser of a vehicle does not own it—she "receives an implied license for the life of the vehicle to operate the vehicle."[16] Uncertainty about the ownership of software-enabled products raises serious questions about the ability of the "owner" of the device to repair or tinker with it or resell it, and disrupts consumers' traditionally settled expectations about ownership of the devices they purchase.

Beyond the relationship between consumers and the products they purchase, copyright law may also complicate the relationship between, and interoperability of, software-enabled products as the demand for compatibility and interoperability increases. While APIs and other

---

[9] *Lexmark* at 552.
[10] Kate Wevers, *Court Rules That Software License Transfers Ownership*, Harvard Journal of Law & Technology (Jolt Digest) (Oct. 12, 2009), *available at* http://jolt.law.harvard.edu/digest/software/vernor-v-autodesk-inc.
[11] *Vernor v. Autodesk, Inc*., 621 F.3d 1102, 1103 (9th Cir. 2010).
[12] Vernor v. Autodesk, 621 F.3d 1102, 1103-1104 (9th Cir. 2010).
[13] *Krause v. Titleserv*, 402 F.3d 119, 124 (2d Cir. 2005).
[14] *Id*.
[15] *Id*.
[16] John Deere Reply Comment, 2015 DMCA proceeding, http://copyright.gov/1201/2015/comments-032715/ class%2021/John_Deere_Class21_1201_2014.pdf at 6.

software used for interoperability and compatibility consist of functional code that should not be copyrightable, the Federal Circuit's recent determination in *Oracle v. Google* (described in detail below) that "the structure, sequence, and organization of the 37 Java API packages [at issue] . . . are entitled to copyright protection,"[17] has introduced considerable uncertainty about that previously settled expectation.

## *B. Allowing Copyright in APIs and Interoperability Software Would Frustrate Innovation and Competition*

When two devices connect and interact—"talk to" each other in some way—they use interfaces to achieve that conversation. These interfaces frequently take the form of APIs:

> APIs are the fundamental building blocks of software programs, used in every software interaction. APIs can be somewhat abstract, so it can help to think of a physical analogue—the Lego® brick. There are many different shapes and colors of Lego® bricks, but they all share one thing in common: the bumps on the top of each brick and the matching holes on the underside. These bumps and holes are the "interface" that allow different bricks to be joined together.[18]

APIs played a fundamental role in the explosive growth of the personal computer and software applications ecosystem. The PC market, though started by IBM, expanded quickly because the open design of its PCs allowed many software and hardware peripheral manufacturers to flourish. IBM's BIOS (Basic Input/Output System) became the industry standard. Other PC manufacturers were able to build computers that interoperated with existing software because they developed their own, compatible BIOSes, developed from an interface specifications known as the BIOS APIs.[19] IBM's challenges to the use were struck down by the courts because the competitors did not copy the underlying BIOS source code, whereas the APIs, were an "uncopyrightable system or method of operation," which was later codified in 17 U.S.C. §102(b).[20]

The PC application market likewise exploded, as IBM's BIOS and another API, provided by the Disk Operating System (DOS), were able to be used by all application developers, who could then be confident that their software would then run on PCs made by all manufacturers. Microsoft's Windows operating system later grew to dominance in large part because its open APIs facilitated the development of a vast ecosystem of applications that worked with the Windows platform. With the advent of the Internet, many more companies developed and offered APIs. Facebook and Twitter each attracted thousands of developers to write software that interoperate with their platforms. The incredible boom in the market for applications—evidenced by the 1.8 and 1.5 million apps available on the Google Play and Apple App stores, respectively—also reveals the explosion in third party innovation spurred through open APIs, as

[17] *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339, 1348 (Fed.Cir.2014).
[18] Brief of Amici Curiae Rackspace US, Inc. et al. at 7, *Oracle America, Inc. v. Google Inc.*, 872 F.Supp.2d 974 (2012) (No. 13-1021, -1022)..
[19] Brief of Amicus Curiae Computer Scientists in Support of Petitioner at 6-8, *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014) (No. 14-1410).
[20] *Id.*, 8.

a large proportion of these apps are designed to interoperate with applications and platforms built by larger tech companies. The modern cloud computing system was created largely relying on a single, compatible interface—allowing for open competition and giving consumers the ability to switch between cloud companies without hesitation.[21]

1. *Oracle v. Google* has upended long-settled expectations about the free use of APIs to interoperate and compete.

*Oracle v. Google*, however, cast serious doubt on the ability of innovators, startups, and others to use and replicate APIs in order to create interoperable products and services. In developing its Android system, Google copied 37 API definitions originally created by Oracle's predecessor for use with the Java programming language. Oracle provides a number of "packages" with Java; these packages are essentially "shortcuts" that "allow programmers to use the pre-written code to build certain functions into their own programs, rather than write their own code to perform those functions from scratch."[22] Each package consists of two parts—"declaring" code, which defines the simple "shortcut" API the programmer uses, and "implementing code," which actually tells the computer how to perform the operations underlying the shortcut.

The District Court applied the merger doctrine to the 7,000 lines of declaring code (which define the API) because "there is only one way to write" them.[23] With respect to the structure, sequence, and organization (SSO) of the APIs, the court ruled in line with earlier cases like *Lotus*, that they constituted "a command structure, a system or method of operation . . . not entitled to copyright protection under Section 102(b) of the Copyright Act."[24] It also recognized that "[d]uplication of the command structure is necessary for interoperability."[25] This decision would have left APIs as they had been historically treated—free for anyone to use to interoperate or re-implement to compete.

On appeal, numerous participants in the software industry strongly endorsed the importance of this conclusion for software innovation. For example, an amicus brief filed by a coalition of start-ups, innovators, and investors explained that interoperability is a "foundational requirement of software innovation."[26] APIs not only make machine-to-machine communication a lot easier (in general APIs save as much as 30% of time for startups to launch a product), but in some cases, programmers "must use the provided APIs because the implementation details are intentionally hidden" to protect copyrightable elements in the software.[27]

Nevertheless, the Federal Circuit reversed, holding that Google could have structured Android differently and could have chosen different ways to express and implement the

---

[21] Stephen J. Vaughan-Nichols, *OpenStack vs. Cloudstack: The Beginning of the Open-Source Cloud Wars,* ZDNet, (Apr. 12, 2012).
[22] *Oracle America, Inc. v. Google Inc.*, 750 F.3d.1339, 1349 (2014).
[23] *Oracle America, Inc. v. Google Inc.*, 872 F.Supp.2d 974, 998 (2012).
[24] *Id.* at 999-1000.
[25] *Id*. at 977.
[26] Brief of Amicus Curiae of Software Innovators, Start-ups, and Investors in Support of Affirmance, at 7. *Oracle America Inc. v. Google Inc.*, 872 F.Supp.2d 974 (2012) (Nos. 2013-1021, -1022).
[27] *Id.*, 7-8.

functionality that it copied.[28] By ruling for the first time that both SSO and the underlying declaring code APIs are copyrightable, the court departed from the settled understanding in the software industry. The decision disrupted the longstanding reliance that the software community had placed upon the expectation that APIs were not copyrightable. This reliance helped the emergence of Apple, Microsoft, Google, Yahoo!, Intel, and Oracle itself, but more recently helped "unleash a tidal wave of innovation in personal and mobile computing, cloud computing, e-commerce, Internet services, and now the emerging Internet of Things.[29] Without free use of software interfaces, many of many inventions would not have been created or realized in the same way, including the home computer, operating systems, programming languages, and cloud computing.[30] Looking forward, the uncertainty surrounding the copyrightability of APIs "adds a new layer of legal risk" to developers' work.[31] As more companies "seize on the *Oracle* finding to threaten copyright litigation against a broad swath of the software industry," individuals and companies will be from deterred from investing in innovation that relies on APIs.[32]

While the impact of *Oracle* may be limited because the Federal Circuit's ruling is not binding on other circuits where future disputes are likely to be resolved, it nevertheless has created uncertainty regarding the copyrightability of software interfaces and the future of interoperable software generally. And as software becomes a part of everyday devices, this uncertainty is carried along with it. It chills innovation and increases the incentives for incumbents in almost any industry to misuse copyright law to try to exclude new entrants and emerging competition.

2. Free reuse and reimplementation of APIs is critical to innovation and competition in the expanding market for software-enabled devices.

APIs have been and are indispensable to interoperability in standalone software platforms and products, and will be equally indispensable to software-enabled consumer devices. Increasingly, software-enabled devices interoperate with and rely on other platforms and devices, often from other developers or manufacturers. For example, in the medical device field, one of the world's leading semiconductor manufacturers has described "[t]he implementation of 'plug and play' communication in the medical device field" as "a reality."[33] Going forward, a "designer will be able to spend more and more of their time and resources on the application and less time on the communication protocols," which would ultimately "lead[] to a more productive, safer and less expensive medical device market."[34] But interoperability relies on public as well as

[28] *Oracle America, Inc. v. Google Inc.*, 750 F.3d.1339, 1368 (2014)
[29] Brief of Amicus Curiae Hewlett-Packard Company, Red Hat, Inc., and Yahoo! Inc. in Support of Petitioner at 14. *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014) (No. 14-410).
[30] Brief of Amici Curiae Computer Scientists in Support of Defendant-Cross Appellant and Affirmance at 4, *Oracle America, Inc. v. Google Inc.*, 872 F.Supp.2d 974 (2012) (Nos. 2013-1021, -1022).
[31] Jeff John Roberts, *Supreme Court's API ruling puts a cloud over software industry*, Fortune.com (June 29, 2015), *available at* http://fortune.com/2015/06/29/supreme-court-oracle-api/.
[32] *Id*.
[33] Andrew Leone, *Medical Device Interoperability: An Embedded Solution*, http:///www.st.com/web/en/resource/technical/document/white_paper/WP_MEDICAL_INTEROPERABILITY_2011_01_01_A.pdf.
[34] *Id.* at 5.

private efforts.[35] The rise of the information economy depended not just on industry participants' decisions to provide open APIs, but on proper judicial demarcation between what is copyrightable and what is not.

A shift to copyright protection for APIs and other functional interoperability systems and methods would inhibit innovators' ability to create and consumers' ability to choose the best technologies, and for those technologies to achieve success in the market. Instead, incumbents with control of the interfaces to their established products may be able to "'trap' the industry in an . . . inferior standard."[36] With consumers and developers "locked in" to its interface, and competitors blocked from offering a competing service using the same interface, one or a handful of incumbent companies can become gatekeepers to the platforms or devices they produce. If a new technology lacks "compatibility with preexisting software" because it cannot obtain a license or it is prohibitively expensive to do so, "the costs and possible limited availability of software will make consumers hesitant to buy the new [technology], even if it is superior."[37] Superior technologies are thus shut out from finding a widespread audience or entering the market altogether.

This chilling effect on innovation can be far-reaching, particularly in secondary markets. Innovators and startups looking to develop software-enabled products in adjacent markets that need to interoperate with or improve upon the underlying platform will face similar issues—they cannot "plug into" the system without paying a licensing fee or creating their own platform from scratch. The requirement to "ask permission" and pay a licensing fee will itself serve to limit new innovation because any marginal increase in the cost to a startup of developing and launching a new product will reduce startup activity and innovation. Once again, existing firms may become gatekeepers, this time to secondary markets. For example, the maker of "smart" locks would be able to select who can make the doorknobs that work with their locks, or deny third parties compatibility altogether, choosing to make and use only their own. This control of innovation in downstream markets prevents innovation and prevents consumers from having access to a wider array of efficient and valuable improvements or complements to their products.

The barriers to entry by innovators and startups created by unnecessary and inappropriate copyright protection are of particular concern to the software interfaces in embedded software, "access to which is essential due to network effects and standardization."[38] Compatibility is necessary to achieve networks "through which users can exchange files [and] standardization of user interfaces [which] prevents user 'lock-in' because users do not have to learn a new user interface in order to switch application programs."[39] This, in turn, gives consumers a broader choice of software and hardware and lowers the barriers to entry to the software industry, which ultimately leads to more innovation and competition, better products and lower prices.

[35] John Palfrey & Urs Gasser, *Interop: the Promise and Perils of Highly Interconnected Systems* 15 (Basic Books 2012).
[36] Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 *Stan. L. Rev.* 1329, 1342 (1986).
[37] Timoty S. Teter, *Merger and the Machine: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Class*, 45 Stan. L. Rev. 1061, 1067, April 1993.
[38] Max Planck Institute for Intellectual Property and Competition Law, *Copyright, Competition and Development*, 7 (Dec. 2013).
[39] Teter at 1067.

To illustrate these far-reaching effects, consider the rapidly-growing home automation industry—specifically, “smart” light bulbs. “Smart” light bulb manufacturers with “penetration pricing strategy or powerful brand name recognition” have the ability to grab a wide audience, who will in turn adopt their interface platforms. In an *Oracle v. Google* world, competitors attempting to enter the marketplace—perhaps with more energy-efficient or more flexible technology—could be stuck with two bad options they would not otherwise face: license the interface by asking permission from the existing bulb companies or build a completely new interface that is incompatible with the pre-existing products. The established companies can choose to effectively shut out new competitors by withholding these software licenses. New competitors are then forced to create new, non-interoperable software interfaces, and consumers are forced to make a longer-term choice of technologies based on who has market power rather than who has the best technology for their particular needs. Stymied by this artificially-created fragmented market, established companies increase their market share in the “smart” light bulb industry with no need to innovate or, really, to compete.

This market control will continue to seep outward to secondary markets. Companies that produce light switches, apps, and additional programs that are meant to interoperate with the light bulbs will need a license to “plug in” to the system. The light bulb companies are then able to pick and choose which downstream companies can interact with their bulbs, or shut them out entirely, favoring their own light switches and apps. These companies are able to use their control in the light bulb market to gain power over secondary markets. This same problem affects the repair market as well. If individuals or third parties are unable to repair the software in their light bulbs without infringing copyright, the bulb companies will also dominate the repair market.

Previously, an existing light bulb company would not have been able to control who could enter the market, what light switch its consumers could use, or who a consumer could call for a repair. In a world where embedded software inhabits our most mundane items, copyright should not be the loophole that bestows such companies with that power. Until now, compatibility “has fostered innovation and competition—two critical policy objectives.”[40] Allowing exclusive rights over software necessary for compatibility would destroy these objectives and ensure that copyright law is used as an “anticompetitive force in the software industry.”[41]

### C. *The Inability of Users to Modify the Software in Devices They Own Hampers Innovation, Prevents Interoperability, Reduces Security, and Undermines Settled Expectations of Ownership*

Since *Chamberlain*, companies have increasingly licensed the software in their products to consumers through click-wrap, shrink-wrap, and other licensing agreements. Licensing rather than sale serves to negate the protections of the first sale doctrine and the computer programs limitation, 17 U.S.C. § 109 and § 117 respectively. As licensees of the copyrighted software in their products, consumers may be lawfully unable to copy, modify, and resell the software in

[40] Michael A. Jacobs, *Copyright and Compatibility*, 30 Jurimetrics J. 91,93 (1989-90).
[41] Brief of Amici Curiae Rackspace US, Inc. et al. at 16, *Oracle*, 872 F.Supp.2d 974 (Nos. 2013-1021, -1022).

their devices. Such limitations come as a big surprise to most consumers and are at odds with long-settled expectations. Owners have almost always enjoyed the right and ability to tinker with, improve, repair and/or sell devices and other property they have purchased. Denying owners of software-embedded devices these usual and expected rights affects far more than the activities of casual hobbyists. Rather, it can significantly frustrate innovation in several ways.

1. The inability of users to modify embedded software prevents user innovation and enhanced interoperability.

First, permissionless innovation extends beyond innovative entrepreneurs and companies to individual owners of products and user communities who wish to broaden or improve their use of those products. Users and user communities sometimes play important roles in innovation.[42] Software embedded in consumer devices can increase functionality and add value, but device manufacturers are not omniscient and cannot take into account all the needs and preferences in the market. Even if they do, a particular target group of consumers might appear to be too small to justify the extra R&D and production costs. Users, however, are not bound by these limitations and may value certain modifications or adjustments enough to justify investing time in making them. Some user innovations improve the user experience of the consumer who makes them and increase the value of the product and embedded software for them. But other innovations are subsequently disseminated within user communities for wider adoption and sometimes even adopted by the manufacturer into subsequent versions of its product. For instance, when a hobbyist shared numerous improvements to his Aibo with other owners, Sony eventually released a development kit to facilitate user modification.[43]

Farm Hack is just one example of a user community where farmers around the world share their innovations with one another. These innovations often incorporate existing technologies and adapt them to more specific situations or activities. Fido, for instance, links text messages with temperature sensors so that farmers can receive real-time alerts about temperature in their greenhouse.[44] This type of incremental innovation is possible largely because farmers are able to tinker with their devices and to use APIs or other interoperability tools between various devices and platforms. To take another example, a consumer might be inclined to modify the software embedded in his refrigerator to link with his online calendar. Another might prefer to develop her own app to interconnect with a home automation system and control temperature or other conditions of her home remotely. In fact, this process is how many entrepreneurs and startups begin: individual users identify limitations in or problems with a particular product or service and develop a solution or add-on features, perhaps first for themselves but then often expanding into a new business and product.

Where licenses prohibit users from accessing or tinkering with the embedded software in their devices, however, individuals may be frustrated in their ability to explore and make these sorts of valuable improvements to their devices and to achieve new interoperability with other

[42] Sometimes an entire industry has begun through user innovation. For instance, the windsurfing equipment industry really began with the footstraps that one hobbyist added onto the board. See Eric von Hippel, Democratizing Innovation 1 (Cambridge: MIT Press 2005).
[43] See Carrier, 190.
[44] See http://farmhack.org/tools/fido-temperature-alarm-sends-text-messages.

devices. Other users also may be denied the benefits of these innovations that could have been shared by the person who developed them. As software-enabled devices continue to proliferate and interoperability becomes increasingly essential, the impact of copyright license restrictions to traditional ownership rights and expectations will become more pronounced and concerning.

2. The inability of users to tinker with or modify embedded software also prevents the investigation and improvement of device security and privacy.

Users, whether regular consumers or sophisticated software researchers, also regularly contribute to the investigation and improvement of the safety, security, and privacy of devices with embedded software. "Hacker" communities have long played an important role in detecting vulnerabilities in software. As embedded software becomes more prevalent, the role of individual users, user communities, and security researchers also will become more significant. Security researchers can identify security flaws and alert the appropriate developers, warn consumers and/or provide their own fixes. Additionally, researchers can identify and lead to the correction of software flaws or design weaknesses that compromise privacy. Finally, they can identify privacy vulnerabilities that the software creator intended but consumers were not aware of and did not agree to. Security research of this sort frequently requires that the software be copied first and then analyzed. All of these activities may require copying, manipulation, or other engagement with the software that may be prohibited by the license under which it is distributed.

The auto industry, an early adopter of embedded software, provides a number of illustrative examples of what security researchers were capable of. In addition to the 2014 Jeep Cherokee recall described earlier, investigators found similar problems on a Tesla Model S, a Chevrolet Corvette, a BMW, and a Mercedes Benz.[45] Analysis of the Wi-Fi-enabled Hello Barbie showcased how cyber-privacy researchers can also spot privacy violations, whether they are deliberate or accidental. The doll's connection to the smartphone app was so unsafe that it was "open to spoofing and interception of all the audio the doll records."[46] Without the ability legally to investigate the software, vulnerabilities like these may not be discovered until they have done irreparable damage.

3. Licensing rather than sale of embedded software undermines settled consumer expectations of ownership and impairs consumers' freedom to sell a device they "bought."

The move toward licensing software to consumers—especially software embedded in consumer devices—rather than selling copies of it has far-reaching consequences for the way the public purchases everyday items. As the Register recently recognized, there remains significant inconsistency and ambiguity in the ways that the first sale doctrine, codified in 17 U.S.C. § 109, and the computer programs limitation, codified in 17 U.S.C. § 117, are applied to embedded software:

> In past rulemaking proceedings, the Register has reviewed case law governing the determination of ownership of a software copy for purposes of section 117 when

[45] Greenberg & Zetter.
[46] *Id*.

> formal title is lacking and/or a license imposes restrictions on the use of the computer program, and has concluded that application of the law can be unclear in some contexts. The Register observed that while *Vernor v. Autodesk, Inc.* and *Krause v. Titleserv, Inc.* may provide some useful guidance in this area, they are "controlling precedent in only two circuits and are inconsistent in their approach."[47]

The uncertainty over ownership of embedded software is growing. John Deere recently asserted to the Copyright Office that, even "[i]n the absence of an express written license in conjunction with the *purchase* of [a vehicle],"[48] the purchaser does not own the copy of the software that helps run the vehicle. Instead, the "*owner* receives an implied license for the life of the vehicle to operate the vehicle, subject to any warranty limitations, disclaimers, or other contractual limitations in the sales contract *or documentation*."[49] It is clear that this is not merely a product of Section 1201 of the DMCA: "Even if TPMs for the vehicle software did not exist, accessing the vehicle software in contravention of these licenses could violate copyright . . . ."[50]

It is no surprise that companies choose to *license* their software embedded in products they otherwise *sell* to consumers—the first sale doctrine may not apply as a result. Indeed, 17 U.S.C. Section 109(d) clearly states that the first sale "privileges prescribed by subsections (a) and (c) do not, unless authorized by the copyright owner, extend to any person who has acquired possession of the copy or phonorecord from the copyright owner, by rental, lease, loan, or otherwise, without acquiring ownership of it." Thus, the increasing prevalence of vendors selling devices with embedded software that is purportedly governed by a software license agreement may serve to negate a purchaser's rights under the first sale doctrine. And while there are some ambiguities in the law regarding whether a software license overrides a sale of the software copy,[51] the lack of clear precedents may exert a chilling effect on consumers' ability and willingness to sell their devices.

Clarifying the application of the first sale doctrine to embedded software would ensure that owners of devices with embedded software have access to at least three statutory benefits. First, "owners" of both a device and its embedded software will be able to take advantage of the first sale doctrine and sell or otherwise transfer both the device and its software under 17 U.S.C. 109(a). However, while this new "ownership" status would allow the device owner to sell the device with the embedded software, it would *not* allow the owner to sell other copies of that software. Second, it appears that the rental, lease, or lending restriction of 17 U.S.C. 109(b)(1)(A) would not apply to a device with embedded software, since subsection 109(b)(1)(B)(i) states that the subsection does not apply to "a computer program which is embodied in a machine or product and which cannot be copied during the ordinary operation or

[47] Section 1201 Rulemaking: Sixth Triennial Proceeding to Determine Exemptions to the Prohibition on Circumvention 160-161, *available at* http://copyright.gov/1201/2015/registers-recommendation.pdf (footnotes omitted).
[48] John Deere Reply Comment at 6.
[49] *Id*. (emphasis added).
[50] *Id*. (emphasis added).
[51] *See., e.g., Vernor v. Autodesk, Inc.*, 621 F.3d 1102 (9th Cir. 2010); *cf. Krause v. Titleserv, Inc.*, 402 F.3d 119, 123 (2d Cir. 2005).

use of the machine or product." And third, statutory owners of a device and its software could dispose of their lawfully owned smart devices without fear of copyright liability.

But even the first sale doctrine may not entirely solve the embedded software problem if the owner of a device wants to tinker with its software and thereafter sell or otherwise transfer his embedded device. As written, 17 U.S.C. Section 117(b) would presumably allow the owner of a device with embedded software to resell the device with an exact copy of that software included. But often the owner of a device would alter the original software to repair or adapt or improve the device, likely resulting in a modified, not original, copy of the software running within the device. Yet Section 117(b) may not allow the owner of that device to unilaterally resell that device with the modified software, since "[a]daptations so prepared may be transferred only with the authorization of the copyright owner." Nor does the software repair provision of Section 117(c) solve this problem, since the repair copy must be "destroyed immediately after the maintenance or repair is completed."

Consumers should not bear this burden of uncertainty. The era of software-enabled devices is now well underway and the Internet of Things is more of a reality every day. A farmer should not face copyright liability for fixing his broken tractor. A parent should not face copyright liability for fixing a broken thermometer. A doctor should not face copyright liability for improving the operation of a critical medical device and sharing the improvements with other physicians or researchers.

## III. NOI Question 4: Whether, and to what extent, legitimate interests or business models for copyright owners and users could be undermined or improved by changes to the copyright law in this area.

Copyright should not prevent the use or replication of APIs or other functional software used for the purpose of interoperability. The legitimate interests of users and the business models of innovators and start-ups, as well as the societal benefits from increased innovation would be improved significantly by eliminating any uncertainty that APIs and other functional software used for interoperability are not copyrightable.

The legitimate interests of users would also benefit from greater clarity and certainty that, in any event, fair use protects the reuse or replication of APIs and other functional software used for interoperability. Fair use also protects tinkering with or copying for the purpose of repairs or testing software embedded in devices. Making such uses is transformative and poses no risk of market substitution or harm to the underlying software work.

The legitimate interests of users also would be improved by, and the Copyright Office should support, the enactment by Congress of provisions similar to those in the You Own Devices Act ("YODA")[52] to ensure that users are treated as owners of the software in their devices and can tinker with, improve, or repair those devices by analyzing, copying, and modifying the software. The Copyright Office should move forward to ensure that appropriate action is taken regarding several of the issues it deferred in its August 2001 Section 104

[52] YODA, H.R. 862, 114th Cong. sec. 2 (2015).

Report.[53] That report determined that no legislative changes were needed at that time to Section 109 to ensure a right of digital first sale. However, it also correctly predicted that ‘‘[t]he time may come when Congress may wish to consider further how to address these concerns,”[54] especially “the operation of the first sale doctrine in the context of works tethered to a particular device.”[55] That time has come; the rapid spread of software-enabled devices and the accompanying emergence of the Internet of Things have highlighted the necessity of first-sale protection for embedded software to benefit users of those devices.

The Section 104 Report also correctly identified the possibility, now a pressing reality, that changes to the copyright laws might eventually be needed to address software licenses and other contractual provisions that impair consumers’ ability to exercise their usual and expected rights over products they own.[56] The “market forces” that the Report hoped might prevent contractual restraints on consumer rights and freedom have not materialized; in fact, just the opposite has happened, as much of the software industry has shifted to a model of licensing embedded and standalone software rather than selling it. Legislative change is needed to address the harm to consumers and innovation from this shift, and the Copyright Office should support such change.

## IV. Conclusion

Engine urges the Copyright Office to strongly support the tremendous opportunities for innovators, entrepreneurs, startups, and users that are being created by the spread of software-enabled products and the Internet of Things. The Office should strive to ensure that copyright law does not restrict the next generations of innovative new products and services that will depend on interoperability with a wide range of existing and emerging platforms and ecosystems to succeed. The innovation that will result will increase participation in the market, enhance competition among products and suppliers, and greatly expand consumer choice and social value.

[53] U.S. Copyright Office, DMCA Section 104 Report (2001).
[54] *Id*. at 96–97.
[55] *Id*. at xvi–xvii.
[56] See *id*. at 162–64.