CONSIDERING A RIGHT TO REPAIR SOFTWARE

Robert W. Gomulkiewicz†

ABSTRACT

The right to repair movement aims to extend the usability of products by allowing a consumer (or a repair professional acting on the consumer's behalf) to fix broken products. Implicitly, the movement's focus has been on hardware—on the right to repair cars, tractors, and phones. But as more and more of the functionality of goods comes from software, it is important to consider whether we need a right to repair software. There are practical challenges to software repair. For example, fixing software is more difficult and treacherous than fixing hardware. Complicating matters further, more and more software is embedded in hardware or runs remotely from the cloud, making it difficult, if not impossible, to repair. A right to repair software would also push deep into conflicts with intellectual property rights because repairing software might infringe a copyright holder's exclusive right to create and distribute derivative works, a patent holder's right to exclude making and using an invention, or a trade secret holder's right to protect valuable information.

This Article attempts to reframe the repair issue as it applies to software in two ways. First, it discusses how a robust conversation about software repair is already well underway as part of the software industry's vigorous debate about the pros and cons of open source software. In other words, right to repair proponents do not need to start a new conversation about the right to repair software; they can and should join the ongoing discussions about open source software. Second, the Article discusses how the most salient issues related to software repair do not involve consumers' ability to fix software bugs but, instead, their ability to get or refuse updates from software developers and to revert to a prior version of the software if the consumer does not like the updated version. Policymakers should focus on these issues as they consider a right to repair software.

TABLE OF CONTENTS

DOI: https://doi.org/10.15779/Z385M6277Z

† Judson Falknor Professor of Law and founding Director of the Intellectual Property Law & Policy Graduate Program, University of Washington School of Law. For useful discussions about issues raised in this Article, I thank Ethan Frederic, Mike Halverson, and Xuan-Thao Nguyen. For excellent research assistance, I thank Alice Men.

I. INTRODUCTION

The right to repair movement wants consumer goods to remain usable for as long as possible.[1] The movement aims to extend the usability of a product by allowing a consumer (or a repair professional acting on the consumer's behalf) to fix broken products. Implicitly, the focus has been on hardware—on the right to repair cars, tractors, and phones.[2] As goods have become smart, however, usability must also take software into account.[3] Proponents of the

1. More fundamentally, the right to repair movement's goals touch on consumer protection, economic fairness, sustainability, and environmentalism. *See* Aaron Perzanowski, *Consumer Perceptions of the Right to Repair*, 96 IND. L.J. 361 (2021); Leah Chan Grinvald & Ofer Tur-Sinai, *Smart Cars, Telematics and Repair*, 54 U. MICH. J.L. REFORM 283, 290-93 (2021); Leah Chan Grinvald & Ofer Tur-Sinai, *Intellectual Property Law and the Right to Repair*, 88 FORDHAM L. REV. 63, 73 (2019); *see also* CEM KANER & DAVID PELS, BAD SOFTWARE: WHAT TO DO WHEN SOFTWARE FAILS (1998) (discussing consumer protection in software purchases).

2. *E.g.*, Elizabeth Chamberlin, *The Elements of Repairable Design*, WALL ST. J., May 31, 2022, at R5.

3. "[T]oday virtually no one can complete a day's work without using a computer. Not only do computers exist on your desk, but a 'computer,' and consequently software, is present in almost every device we use. . . Software is pervasive . . ." GLENFORD J. MYERS, COREY SANDLER & TOM BADGETT, THE ART OF SOFTWARE TESTING 1 (3d ed. 2012); *see* Chris Jay Hoofnagle, Aniket Kesari & Aaron Perzanowski, *The Tethered Economy*, 87 GEO. WASH. L. REV. 783 (2019); Stacy Ann Elvy, *Hybrid Transactions and the INTERNET of Things: Goods, Services, or*

right to repair have addressed this reality to an extent. For example, they have proposed a right to access security code information and diagnostic software, as well as amendments[4] to the Digital Millennium Copyright Act's prohibitions on tampering with technical protection measures.[5] But as more and more of the functionality of goods comes from software, it is important to consider a fundamental question: do we need a right to repair software?

To answer that question, this Article examines the repair landscape in the software industry. Understanding this landscape should be useful to policymakers and regulators as they evaluate whether legislative or regulatory intervention is needed and, if so, what its focus should be.[6] Acting carefully is particularly important for software repairs for at least two reasons.

First, there are practical challenges to providing a right to repair software. Fixing software is often more difficult and treacherous than fixing hardware. Indeed, fixing one software bug can lead to many other bugs.[7] To complicate matters further, more and more software either comes embedded in hardware or runs remotely from the cloud. Software distributed in these ways is particularly difficult (if not impossible) to access in a manner that would allow a consumer, or even an independent software programmer, to repair the software.

Second, a right to repair software would push deep into conflicts with intellectual property rights.[8] Repairing software might infringe a copyright

Software, 74 WASH. & LEE L. REV. 77 (2017). Some consumers do not appreciate the trend toward so-called "smart" devices. *See* Justin Pot, *Stressed By Our Devices*, WALL ST. J., Apr. 16-17, 2022, at D1, D6 ("As so-called 'smart-tech' becomes more connected, it's more prone to unexpected bugs and glitches. That's why skeptics are turning to 'dumber' appliances and gear.").

4. *See* 37 C.F.R. § 201.40 (2019).

5. *See* 17 U.S.C. § 1201; Lindsey Barrington, Comment, *Manufacturers Beware of Right to Repair: An Analysis of the Resurgence of Right to Repair & the Legal Consequences of Third-Party Access to Embedded Software in the 'Internet of Things' Era*, 20 SUSTAINABLE DEV. L. & POL'Y 24, 25-26 (2020).

6. *See generally* Paul Ohm & Blake Reid, *Regulating Software When Everything Has Software*, 84 GEO. WASH. L. REV. 1672, 1686-89 (2016) (highlighting the "complex and interconnected suite of policy issues, values, and law that will often arise when an agency tries to regulate hardware that has shifted to incorporate software" and noting that there are sure to be "growing pains" and "unintended consequences" from any new regulations as a result).

7. As discussed in Section V, *infra*, this concern is reflected in open source software licenses which insist on disclaimers of warranty as the trade-off for freedom to tinker.

8. *See generally* Leah Chan Grinvald & Ofer Tur-Sinai, *Intellectual Property Law and the Right to Repair*, 88 FORDHAM L. REV. 63, 105 (2019). The recent FTC report on the right to repair noted that "A full discussion of the interplay between intellectual property and repair is beyond the scope of this report." FTC, NIXING THE FIX: AN FTC REPORT TO CONGRESS ON REPAIR RESTRICTIONS 26 (2019) [hereinafter "FTC REPORT"]. "[A]ny action taken by industry or regulators to enable independent repair should seek input from such entities and other

holder's exclusive right to create and distribute derivative works, a patent holder's right to exclude making and using an invention, or a trade secret holder's right to protect valuable information. For example, many software developers protect their source code as a crown jewel trade secret, so mandating access to that code could jeopardize a valuable business asset.

Given the intellectual property law challenges presented by a right to repair software, this Article discusses how a right to repair software could be accommodated in intellectual property law, especially copyright law. Other commentators have explored changes to the Digital Millennium Copyright Act that would be congenial to repairing smart goods.[9] This Article builds on that work by examining how Congress could accommodate software repairs by amending the Copyright Act's § 117, which already provides for computer hardware repairs.

However, even though policymakers *could* amend intellectual property law, *should* they do so? To answer that question, this Article highlights non-legislative avenues for software repair,[10] particularly how software repair fits into existing software licensing practices. The software industry already provides multiple avenues for software repair.[11] Software developers regularly supply bug fixes, security patches, and a variety of other updates to their users, often at no charge.[12] In addition, software developers license their source code through various channels to enable software repairs. Most prominently, developers of open source software embrace and, indeed, extoll the right to repair software. Even when source code licensing is not available, courts have consistently recognized a fair use right under copyright law when customers reverse engineer software to discover and use uncopyrightable ideas or information.[13] Moreover, end user licenses agreements (EULAs) for software

stakeholders and be mindful of existing law and policy supporting IP protection." FTC REPORT, at 53-54.

9. *See, e.g.*, Madison Bower, Comment, *Keeping the DCMA Away From Functional Use*, 35 BERKELEY TECH. L.J. 1067 (2020).

10. According to the FTC: "[S]elf-regulation can help address concerns about repair restrictions in discrete markets. But, no industry sector other than the automotive industry has worked to open repair markets through a self-regulatory framework. Ways to stimulate self-regulation in markets beyond the automotive sector, however, merit further consideration." FTC REPORT, at 46.

11. Moreover, 17 U.S.C. § 117(c) provides that an owner or lessee of a computer may make a copy of a computer program for purposes of maintenance or repair of the computer. *But see* MAI Sys. Corp. v. Peak Comput., Inc., 991 F.2d 511 (9th Cir. 1993) (repair activities in this case are not permitted under copyright law).

12. *See* Brian X. Chen, *Yes, you can make your tech survive obsolescence*, SEATTLE TIMES, Mar. 26, 2022, at A12-13.

13. Right to repair proponents seem to equate restrictions on reverse engineering with restrictions on repair. *See* FTC REPORT, at 24.

could also affirmatively address repair and this Article suggests reasons why taking a proactive approach to providing information about software repair in EULAs might make good sense.

Finally, this Article attempts to reframe the repair issue as it applies to software in two ways. First, a robust conversation about software repair is already well underway because it is part of the software industry's vigorous debate about the pros and cons of open source software. In other words, right to repair proponents do not need to start a new conversation about the right to repair software, they can and should join the ongoing discussions about open source software. Second, the most salient issues related to software repair do not involve consumers' ability to fix software bugs. Instead, they involve a consumer's ability to get and refuse updates from software developers and to revert to a prior version of the software if the consumer does not like the updated version. With this reframing in mind, policymakers and regulators can evaluate more clearly and precisely whether legislative or regulatory intervention is warranted, or whether it is best to leave matters to competition in the market.

Following this Introduction, this Article proceeds in several sections. Section II provides a basic background on software forms, storage, and distribution. Section III provides a primer on legal protection for software. Building on that background, Section IV explores the landscape of software repair and links that landscape to legal protection for software. Section V then considers potential new legislative and non-legislative approaches to software repair. Section VI provides some concluding observations and reflections.

II. SOFTWARE FORMS, STORAGE, AND DISTRIBUTION

Today, software is common and ubiquitous, so it is hard to believe that not long ago, software was nearly invisible to us. In the early 1950s, *Fortune* magazine published an article titled "Office Robots," which was one of the first pieces in the popular press to discuss computers.[14] The article focused on computer hardware, however, not software. The "software" nomenclature came into general usage around 1960[15] and the media finally began to recognize the emergence of a discrete software industry in the early 1980s.[16] By 1984 a

14. *Office Robots*, FORTUNE, Jan. 1952, at 87.

15. *See* FREDRICK P. BROOKS, THE MYTHICAL MAN MONTH: ESSAYS ON SOFTWARE ENGINEERING 4 (1975) (this book is considered one of the classic works on software development).

16. *See* MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY (2004); MICHAEL A. CUSUMANO, THE BUSINESS OF SOFTWARE: WHAT EVERY MANAGER, PROGRAMMER, AND

Business Week headline proclaimed software "The New Driving Force" of the U.S. economy.[17]

A. FORMS OF SOFTWARE

Software consists of statements or instructions that are executed by a computer to produce a certain result.[18] Or, to put it another way, software is digital information that performs a function on a computer. A software developer would say that software comes in two basic forms: source code and object code.[19] Source code refers to the code written by software programmers in a computer language such as BASIC, C/C++, or Java. Source code is human-readable code—it can be understood by any programmer proficient in the language in which it is written.

Object code is derived from source code using a software tool called a compiler. Object code consists of a series of ones and zeros, so it is sometimes called binary code. Object code is stored on a computer-readable medium, such as a hard drive or CD-ROM, and executes (i.e., runs) on the computer hardware. Because of this, it is sometimes referred to as executable code or machine-readable code.

You can also think of software from the user's point of view. The software's visual displays and its ability to accept user input (through keyboard, mouse, touch, voice, etc.) is known as the user interface. Another aspect of software is the user's experience or the service it provides. Software publishers sometimes call this "software as a service." Software code in this sense remains largely invisible to the user (at least so long as the software is working properly). The world is on the verge of the computer revolution foreseen by World Wide Web creator Tim Berners-Lee, where computers, the network, and the software that drives them are invisible to the user.[20]

ENTREPRENEUR MUST KNOW TO THRIVE AND SURVIVE IN GOOD TIMES AND BAD (2004); *see also* TRACY KIDDER, THE SOUL OF A NEW MACHINE (1981) (Pulitzer Prize winning popular book focusing on computer and software development).

17. *Software: The New Driving Force*, BUS. WK., Feb. 27, 1984, at 54.

18. *See* 17 U.S.C. § 101.

19. *See* ROBERT W. GOMULKIEWICZ, SOFTWARE LAW AND ITS APPLICATION 4-7 (2d ed. 2018). Code can take other forms as well, but it is sufficient to focus on source and object code for purposes of this Article.

20. *See* Tim Berners-Lee, James Hendler & Ora Lassila, *The Semantic Web*, SCI. AM., May 17, 2001.

B. SOFTWARE STORAGE AND DISTRIBUTION: FROM PUNCH CARDS TO THE CLOUD

Software needs a way to provide its instructions to the computer hardware. To do this, software instructions must be stored on some type of media and then retrieved by the hardware at the opportune time. At one point, software programs were stored on punch tapes or stacks of punch cards that were fed into the computer.[21] By the 1980s, object code was stored and distributed on diskettes (8-inch and then smaller 5 ¼ and 3 ½ inch floppy disks).[22] By the 1990s, object code was often stored and distributed on computer hard drives[23] as well as on CD-ROMs (compact read only memory), which had greater storage capacity than diskettes.[24] Today, many devices—from heart monitors to refrigerators—have become platforms capable of storing and running software. Software is distributed by hard wiring, burning, or otherwise embedding object code into the structure of these devices.[25] Increasingly, however, software is not stored on a local device but is accessed and run from software stored remotely, including the web of computer servers that we call "the cloud."[26]

III. LEGAL PROTECTION FOR SOFTWARE

The United States does not have a *sui generis* law that protects software. Instead, software developers rely on copyright, patent, trade secret, and contract law.[27] Copyright law protects software in its source and object code forms as well as the visual displays that it generates.[28] Copyright law gives the software developer the exclusive right to copy, distribute, and create derivative works of the software.[29] Software-related inventions may be protected by

21. *See* MICHAEL J. HALVORSON, CODE NATION: PERSONAL COMPUTING AND THE LEARN TO PROGRAM MOVEMENT IN AMERICA 65-66, 79-80 (2020).

22. *See generally* EDWARD TEJA, THE DESIGNER'S GUIDE TO DISK DRIVES (1985).

23. *See* THOMAS HAIGH & PAUL E. CERUZZI, A NEW HISTORY OF MODERN COMPUTING 222 (2021) ("Even the slowest hard drives transferred data far more rapidly than floppy disk drives and, for most users, could hold a complete collection of programs and working data.").

24. *Id.* at 304-05.

25. Sometimes referred to as "firmware."

26. HAIGH & CERUZZI, *supra* note 23, at 360-71.

27. *See* GOMULKIEWICZ, *supra* note 19, at 7-12.

28. *See* Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183 (2021); Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994) (visual displays); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240 (3d Cir. 1983) (source and object code).

29. 17 U.S.C. § 106.

patent law,[30] although granting patents for software is currently a hotly debated topic in patent law.[31] A patent gives the patentee the right to exclude others from making, using, or selling products that embody the patented invention.[32] Software source code that is guarded from discovery using reasonable measures may be protected under trade secret law.[33] And finally, contracts work in tandem with intellectual property laws,[34] such as by contributing to the measures required for trade secret protection.[35]

IV. THE LANDSCAPE OF SOFTWARE REPAIR

When software developers discuss software repairs, they talk about fixing "bugs."[36] All software has bugs, even the highest quality code.[37] So, in a sense, software is always broken and in need of repair. This section provides an overview of how software repair has unfolded in the software industry. Some things have changed but many things have remained the same. To provide some context, however, it is useful to present a few fundamentals about software development and repair.

A. SOME FUNDAMENTALS OF SOFTWARE DEVELOPMENT AND REPAIR

A fundamental principle of software development is that the software program does not stop changing when it is delivered to the customer. Programmers call this program maintenance, which includes adding new

30. *See* Alice Corp. v. CLS Bank Int'l, 134 S.Ct. 2347 (2014); *see generally* 2019 Revised Patent Subject Matter Eligibility Guidance, 84 Fed. Reg. (Oct. 18, 2019).

31. *See* Andrew A. Toole & Nicholas A. Pairolero, *Adjusting to Alice: USPTO Outcomes After Alice Corp. v. CLS Bank*, *in* OFFICE OF THE CHIEF ECONOMIST IP DATA HIGHLIGHTS 1, 5–6 (Apr. 23, 2020); Mark A. Perry & Jaysen S. Chung, *Alice at Six: Patent Eligibility Comes of Age*, 20 CHI.-KENT J. INTELL. PROP. 64, 73 (2021).

32. *See* Impression Prods., Inc. v. Lexmark Int'l, Inc., 137 S. Ct. 1523 (2017).

33. *See* GOMULKIEWICZ, *supra* note 19, at 10-11.

34. *See generally* Raymond T. Nimmer, *Breaking Barriers: The Relation Between Contract and Intellectual Property Law*, 13 BERKELEY TECH. L.J. 827 (1998). Historically, contracts have always been a significant part of the intellectual property protection equation. *See* Robert W. Gomulkiewicz, *Contracts Mattered as Much as Copyrights*, 66 J. COPYRIGHT SOC'Y U.S.A. 441 (2019).

35. *See* Robert W. Gomulkiewicz, *Fostering the Business of Innovation: The Untold Story of* Bowers v. Baystate Technologies, 7 WASH. J. L. TECH. & ARTS 445, 450-51 (2012).

36. Popularization of the word "bug" for software defects is often traced to computer pioneer Grace Hopper. When Hopper was released from active military duty, she joined the Harvard faculty at the Computation Laboratory where she continued her work on the Mark II and Mark III computers. Operators traced an error in the Mark II to a moth trapped in a relay. This bug was carefully removed and taped to the logbook. The logbook can be found in the Smithsonian's National Museum of American History.

37. *See generally* MYERS ET AL., *supra* note 3, at 5-18; CEM KANER, JACK FALK & HUNG Q. NGUYEN, TESTING COMPUTER SOFTWARE 17-54 (2d ed. 1999).

functions and fixing defects.[38] The cost and complexity of software program maintenance are related to the size of the program and the number of users—the bigger the program and the more users, the larger the number of defects that will be found and in need of repair.[39] A distinct challenge with program maintenance is that fixing a bug creates a substantial chance of introducing another bug. Often this means that fixing a bug is two steps forward and one step back.[40] But as a large software program evolves, the cumulative effect of all the changes tends to degrade the structure of the program so that, as time goes by, the software becomes less and less well ordered. At some point, repairing a defect can become one step forward and one step back.[41]

A corollary to this fundamental principle of software repair is that fixing software is different and more complex than repairing hardware.[42] As Frederick Brooks explains in his classic work on software development, THE MYTHICAL MAN MONTH:

> Software entities are more complex for their size than perhaps any other human construct, because no two parts are alike. . . In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound. Digital computers are themselves more complex than most things people build; they have very large numbers of states. This makes conceiving, describing, and testing them hard. Software systems have orders of magnitude more states than computers do. . . . Many of the classic problems of developing software derive from this essential complexity and its nonlinear increases with size.[43]

B. SOFTWARE REPAIR AVENUES

1. Early Source Code Licensing

In the early days of software development, programmers shared their source code with other programmers to enable repairs and other modifications.[44] Sometimes the right to repair the software was captured in a

38. BROOKS, *supra* note 15, at 120.

39. *Id.* at 121.

40. *Id.* at 122.

41. *Id.* at 122-23. *See also* PASCAL ZACHERY, SHOW STOPPER!: THE BREAKNECK RACE TO CREATE WINDOWS NT AND THE NEXT GENERATION AT MICROSOFT (1994) (describing Microsoft's race against bugs in releasing its next generation operating system, Windows NT, including "show stopper" bugs revealed late in the development process).

42. BROOKS, *supra* note 15, at 120-21.

43. *Id.* at 183.

44. *See* ROBERT W. GOMULKIEWICZ, XUAN-THAO N. NGUYEN & DANIELLE M. CONWAY, LICENSING INTELLECTUAL PROPERTY: LAW AND APPLICATION 458-59 (4th ed. 2018).

written license contract, but often the right arose by implication, course of dealing, or custom. As time went by, many businesses acquired software to improve their operations, but most of the customers did not have the expertise to repair the software themselves.[45] Consequently, a customer might enter into a service contract with the developer of the software to maintain and repair the code, or a customer might hire an independent contractor who specialized in software repair. If a customer decided to hire a contractor to make software repairs, then the customer (or the contractor) had to acquire the source code from the original developer along with a license to copy and create derivative works of the software.

2. Object Code Patches

By the 1980s, software had become a mass market product. Some businesses continued to license source code from the original software developer for repair purposes. However, many customers did not have the inclination or the resources to manage software repairs, so software developers established channels to provide their bug fixes to customers. Sometimes this took the form of a maintenance contract, where the developer agreed to provide bug fixes in object code form directly to a customer for a certain period of time. But other channels for repair emerged as well. For example, companies such as Electronic Data Systems and Perot Systems became experts in hosting and maintaining software infrastructure for large customers.[46] These companies took on the responsibility of either using source code to repair the customer's software or acquiring and installing object code patches. Value-added-resellers (VARs),[47] systems integrators, and a variety of software services firms[48] provided maintenance and repair services for smaller end users.

As customers began to connect to the internet, software companies used that channel to provide repairs directly to customers. Initially, object code patches were simply available for download from the software developer's website. Then developers began to "push" object code to customers—the customer could choose to install the repair code with the click of a button or, to the consternation of some, the repair code just installed automatically.

45. Expertise to repair software is different, of course, than basic programming skills. *See generally* HALVORSON, *supra* note 21.

46. *See generally* JEFFERY R. YOST, MAKING IT WORK: A HISTORY OF THE COMPUTER SERVICES INDUSTRY (2017).

47. *E.g.*, Ariz. Retail Sys., Inc. v. Software Link, Inc., 831 F. Supp. 759 (D. Ariz. 1993).

48. *E.g.*, Asset Mktg. Sys., Inc. v. Gagnon, 542 F. 3d 748 (9th Cir. 2008) (independent contractor providing software services d/b/a "Mister Computer").

3. Software Subscriptions

In recent times, many software developers have moved away from a business model that emphasizes distributing object code copies of their software. Now, many developers use a subscription model where software is provided remotely via the cloud.[49] In a subscription model, the customer automatically receives the most up-to-date software available during the subscription period, so repairs are simply part of the subscription's value proposition.[50]

4. Confidential Source Code Licensing and Reverse Engineering

As discussed above, software developers often license their source code. Many software developers hold their source code as a trade secret, which adds complexity and sensitivity to source code licenses. The license contract must contain features of both a copyright and trade secret license, including the delineation of measures to protect the secrecy of the source code. And, as a practical matter, the more licenses the developer grants, the greater the risk that trade secrets will be lost. Thus, even though confidential source code licensing is common, it is not ubiquitous.

So, what happens if a customer wants but cannot get a source code license for repairs? The customer (or its contractor) can reverse engineer the object code to discover the source code.[51] This is accomplished by running the object code through a software tool that reverses the compilation process, taking the software from machine-readable object code back to human-readable source code.

The Supreme Court has characterized reverse engineering as an "essential part of innovation."[52] Trade secret law considers reverse engineering a proper means of discovering information.[53] Several courts have ruled that making intermediate copies of software to uncover unprotectable ideas may amount to a defensible "fair use" under the Copyright Act.[54]

---

49. *See, e.g.*, Sarah E. Needleman, *Sony Videogame Unit Plan To Combine Two Services*, WALL ST. J., Mar. 30, 2022, at B4; Matt Day, *Office 365 Beats Sales of Old Office*, SEATTLE TIMES, July 21, 2017, at B15.

50. HAIGH & CERUZZI, *supra* note 23, at 383-84.

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

52. Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 160 (1989).

53. *See* RESTATEMENT (FIRST) OF TORTS § 757 cmt. f; RESTATEMENT (THIRD) OF UNFAIR COMPETITION § 43; UNIFORM TRADE SECRETS ACT § 1, official cmt.

54. *See, e.g.*, Sony Comput., Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000); Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832 (Fed. Cir. 1992).

However, software developers often distribute software in object code form under a license contract that prohibits reverse engineering. Thus, even though reverse engineering may not infringe a copyright, it may breach a contract. These contractual prohibitions on reverse engineering have inspired a great deal of scholarly scorn[55] but have been largely upheld by courts.[56] Even so, courts ensure that the software developer take the necessary steps to form an enforceable contract and have noted that the damages for breach of contract in many instances would be *de minimis*.[57]

Moreover, as a practical matter, reverse engineering object code to discover source code can be very time consuming and may not yield that much useful information.[58] So, even though reverse engineering may be possible and legal, it may not yield the information that the customer actually needs to repair the software.

5. Open Source Software

In common parlance, "open source" simply refers to a philosophy of freely sharing ideas, research, or materials. In the software industry, however, open source refers to a specific software development model.[59] In that model, a programmer creates some software, posts the source code on the internet, and a community of developers grow up around the software as the community tinkers with the code.[60] As the discussion, *supra*, illustrates, software is protected by intellectual property law, so an intellectual property license is

55. *See, e.g.*, David A. Rice, *Copyright and Contract: Preemption After* Bowers v. Baystate, 9 ROGER WILLIAMS U. L. REV. 595, 644 (2004); Jonathan Wilson, Comment, *Can a Copyright Holder Prevent Reverse Engineering - The Federal Circuit Court Holds that the Federal Copyright Act Does Not Preempt No Reverse Engineering Clauses*, 8 COMPUT. L. REV. & TECH. J. 467 (2004); Sara Bressman, Comment, *Restricting Reverse Engineering through Shrink-Wrap Licenses:* Bowers v. Baystate Technologies, Inc., 9 B.U. J. SCI. & TECH. L. 185 (2003).

56. *See* Gomulkiewicz, *supra* note 35.

57. *See* Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1326 (Fed. Cir. 2003), *cert. denied*, 123 S. Ct. 2588 (2013).

58. *See* Andrew Johnson-Laird, *Reverse Engineering in the Real World*, 19 U. DAYTON L. REV. 843 (1994); Andrew Johnson-Laird, *Reverse Engineering of Software: Separating Legal Mythology from Actual Technology*, 5 SOFTWARE L.J. 331, 342-43 (1992).

59. Some programmers prefer to use the term "free software" because it connotes *freedom* rather than simply open access and liberal use. *See* RICHARD M. STALLMAN, *Why "Free Software" is Better than "Open Source," in* FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 55-56 (Joshua Gay ed. 2002). The practical distinction is explored, *infra*, in Section V, as it relates to differences in requiring the sharing of derivative works. For purposes of this Article, however, I will use the term "open source" to encompass "free software" except in instances where the two approaches diverge.

60. *See generally* CHRISTOPHER M. KELTY, TWO BITS: THE CULTURAL SIGNIFICANCE OF FREE SOFTWARE (2008); STEVEN WEBER, THE SUCCESS OF OPEN SOURCE (2004).

needed to facilitate open source software development.[61] Specifically, a license must grant unfettered access to the software source code, an unlimited right to copy the software, and permission to create and distribute derivative works of the software.[62]

The open source software movement has its roots in the hobbyist and scientific communities where software developers routinely distribute source code so they can collaborate on projects.[63] The principles of free modification and distribution of source code were institutionalized in 1985 by Richard Stallman who founded the Free Software Foundation and created the General Public License (GPL) to distribute his software.[64] The open source movement burst onto the public stage in 1998 when Netscape announced that it would license the source code of its popular Navigator web browser (which was re-named "Mozilla" and then "Firefox").[65] Subsequently, open source programs such as the Linux kernel and Apache web server became common software technologies and large computer companies such as IBM and Intel embraced open source software. Over time, the open source movement spread to

61. *See* HEATHER J. MEEKER, THE OPEN SOURCE ALTERNATIVE: UNDERSTANDING RISKS AND LEVERAGING OPPORTUNITIES (2008); LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW (2005); Robert W. Gomulkiewicz, *How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B*, 36 HOUS. L. REV. 179 (1999). An organization called the Open Source Initiative (OSI) certifies licenses as "open source" if the license complies with its Open Source Definition. While OSI has approved dozens of licenses, in practice most programmers use either the Free Software Foundation's *General Public License* (the "GPL") or some variation of a license known as the *BSD License* (so named because it was first used to license U.C. Berkeley's variant of UNIX, the Berkeley Software Distribution). *See generally* Robert W. Gomulkiewicz, *Open Source License Proliferation: Helpful Diversity or Hopeless Confusion*, 30 WASH. U. J. L. & POLC'Y 261 (2009); Robert W. Gomulkiewicz, *De-Bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75 (2002).

62. *See generally* Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443 (2005); Josh Lerner & Jean Tirole, *The Scope of Open Source Licensing*, 21 J.L. ECON. & ORG. 20 (2005); Daniel B. Ravicher, *Facilitating Collaborative Development: The Enforceability of Mass-Market Public Software Licenses*, 5 VA. J.L. & TECH. 11 (2000).

63. *See* Gomulkiewicz*, How Copyleft Uses License Rights, supra* note61, at 182-83.

64. *See* Robert W. Gomulkiewicz, *General Public License 3.0: Hacking the Free Software Movement's Constitution*, 42 HOUS. L. REV. 1015 (2005).

65. *See* Jim Hamerly, Tom Paquin & Susan Walton, *Freeing the Source: The Story of Mozilla*, *in* OPEN SOURCES 197 (Chris DiBona et al. eds. 1999).

governments around the world[66] and captured even early skeptics[67] such as Microsoft, which is now heavily involved in supporting open source software.[68]

In open source software development, tinkering goes far beyond software repair, of course. However, fixing bugs is always specifically mentioned as one of the core purposes and comparative advantages of open source development. In open source development, a community of programmers from around the world can constantly identify problems and create patches that fix the problems. As prominent hacker Eric Raymond puts it: "given enough eyeballs, all bugs are shallow."[69] Thus, the basic goals of the right to repair movement coincide with the goals of the open source software movement.

V. LEGISLATING A RIGHT TO REPAIR SOFTWARE?

A. FRAMING SOFTWARE REPAIR LEGISLATION

As noted, a right to repair software would touch on a copyright holder's exclusive right to create and distribute derivative works, a patent holder's right to exclude making and using an invention, and a trade secret holder's right to protect valuable information.[70] Consequently, state legislation providing for a

66. *See* European Commission, *Open Source Software 2020-2023: Think Open* (Oct. 21, 2020); European Commission, *The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy* (2021) (Final Study Report); Meaghan Tobin, *China wants to build an open source ecosystem to rival GitHub*, REST OF WORLD (Jan. 19, 2021), https://restofworld.org/2021/china-gitee-to-rival-github/; Lauren Dobberstein, *Beijing wants to level up China's software industry, with an emphasis on FOSS*, REGISTER (Dec. 1, 2021), https://www.theregister.com/2021/12/01/china_five_year_software_plan/.

67. *See* Joseph Scott Miller, *Allchin's Folly: Exploding Some Myths about Open Source Software*, 20 CARDOZO ARTS & ENT. L.J. 491 (2002).

68. *See* Steven J. Vaughan-Nichols, *Open source has won, and Microsoft has surrendered*, COMPUTERWORLD (Nov. 28, 2016), https://www.computerworld.com/article/3144063/open-source-has-won-and-microsoft-has-surrendered.html; Asha Barbaschow, *Why open source is so important to Microsoft*, ZDNET (Feb. 28, 2018), https://www.zdnet.com/article/why-open-source-is-so-important-to-microsoft/.

69. ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR (1999). Raymond calls this "Linus's Law" in honor Linus Torvalds who developed the popular Linux software. However, some people question whether open source development's "many eyeballs" approach is better at debugging software than stringent review by a smaller group of developers. *See* ROBERT L. GLASS, FACTS AND FALLACIES OF SOFTWARE ENGINEERING (2003); Eric Schmidt & Frank Long, *Protect Open-Source Software*, WALL. ST. J., Jan. 28, 2022, at A15 (discussing security vulnerabilities in open source software and proposing a federal government center to facilitate improved security).

70. The MODEL RIGHT TO REPAIR LAW § 5(a) requires disclosure of trade secrets to the extent "necessary to provide documentation, parts, and tools on fair and reasonable terms."

right to repair may be preempted by federal copyright and patent law.[71] Trade secret law might be different because it is still largely state law, but the state legislature would have to reconcile and integrate the right to repair legislation with its trade secret statute[72] and take into account the federal Defend Trade Secrets Act, which operates concurrently with state trade secret law.[73]

With that said, of course, Congress could amend copyright and patent law to account for a right to repair, as some have proposed. For copyright, Congress has a prior model[74] in 17 U.S.C. 117(c)-(d).[75] That provision, focused on computer hardware, could be revised to provide for a right to software repair as follows:

> (x) Software Maintenance or Repair.—Notwithstanding the provisions of section 106, it is not an infringement for the owner, lessee, or licensee of a copy of a computer program to (i) copy or authorize the copying or (ii) make or authorize the making of a derivative work, solely for purposes of repair of that computer program, if—
>
> (1) the new copy or derivative work is for use only by the owner, lessee or licensee; and
>
> (2) any new copy or derivative work is used in no other manner.

However, some proposed right to repair laws do not require disclosure of trade secrets. *See, e.g.*, H.R. 1649, 29th Leg., Reg. Sess. § 8 (Haw. 2018); H.R. 3030, 100th Gen. Assemb., Reg. Sess. § 35 (Ill. 2017). *See also* Ofer Tur-Sinai & Leah Chan Grinvald, *Repairing Medical Equipment in Time of Pandemic*, 52 SETON HALL L. REV. 484 (2021) (discussing trade secrets in the context of the Critical Medical Infrastructure Right to Repair Act).

71. *See* Bonito Boats, Inc. v. Thundercraft Boats, Inc., 89 U.S. 141 (1989). *See generally* Leah Chan Grinvald & Ofer Tur-Sinai, *Intellectual Property Law and the Right to Repair*, 88 FORDHAM L. REV. 63, 105 (2019); Nicholas A. Mirr, Comment, *Defending the Right to Repair: An Argument for Federal Legislation Guaranteeing the Right to Repair*, 105 IOWA L. REV. 2393 (2020).

72. States do this, for example, with their freedom of information and public records laws which mandate transparency but account for trade secrets. *See* John Delaney, Comment, *Safeguarding Washington's Trade Secrets: Protecting Businesses from Public Records Requests*, 92 WASH. L. REV. 1905 (2017). Another example is California's law prohibiting covenants-not-to-compete which has to be reconciled with its trade secret statute. *See* Robert W. Gomulkiewicz, *Leaky Covenants-Not-to-Compete as the Legal Infrastructure for Innovation*, 49 U.C. DAVIS L. REV. 251, 291-94 (2015).

73. 18 U.S.C. §§ 1831-39 (2016).

74. Congress could also look to the example of the European Union's software directive, specifically Articles 4(1) and 5(1). Directive 2009/24/EC of the European Parliament and of the Council on the legal protection of computer programs, 2009 O.J. (L 111).

75. *See generally* Alan Galloway, Comment, *Preserving Competition for Computer Maintenance in the DMCA Era: 17 U.S.C. Sec. 117(c) and Sec. 1201(a)(1) after* StorageTek, 22 BERKELEY TECH. L.J. 293 (2007).

> (z) Definitions.—For purposes of this section—
>
> (1) the "repair" of a computer program is the restoring of the computer program to the state of working in accordance with its original specifications and any changes to those specifications authorized by the author of the computer program.

Patent law has developed a right to repair doctrine as part of its exhaustion doctrine.[76] Patent's exhaustion doctrine (in contrast to copyright's) is based on common law rather than federal statute.[77] However, there is no reason why Congress could not add a right to repair into the Patent Act.

If Congress decides to permit a right to repair software, then it should also consider addressing the availability of repair information. For software repairs, this would mean that a customer could distribute its bug fixes and security patches for anyone to use. This distribution touches on a right granted by the Copyright Act—the exclusive right to control the distribution of a work, including any derivative work. By touching on the distribution right as well as the right to create derivative works, the legislation would push deeply into fundamental copyrights.[78] In addition, as described in the next Section, lawmakers would be wading into a heated debate in the open source community about whether distribution of derivative works should be mandatory or voluntary.

B. REFRAINING FROM SOFTWARE REPAIR LEGISLATION?

Congress *could* act—but *should* Congress act? Arguably, the software industry is well down the road in considering a right to repair software because software repair has been subsumed in the ongoing evaluation of the pros and cons of open source software. Indeed, state and federal governments in the United States and overseas have been deeply involved in the conversation

76. *See* Wilson v. Simpson, 50 U.S. 109, 123 (1850); Aro Mfg. Co. v. Convertible Top Replacement Co., 365 U.S. 336, 345 (1961); Jazz Photo Corp. v. Int'l Trade Comm'n, 264 F.3d 1094, 1102 (Fed. Cir. 2001); Husky Injection Molding Sys. v. R&D Tool & Eng'g Co., 291 F.3d 780, 784-85 (Fed. Cir. 2002). *See generally* Mark D. Janis, *A Tale of the Apocryphal Axe: Repair, Reconstruction, and the Implied License in Intellectual Property Law*, 58 MD. L. REV. 423, 427 (1999); Natali Richter, Comment, *"Substantial Embodiments" and "Readily Replaceable Parts": A Contemporary Understanding of the Doctrine of Permissible Repair*, 59 U. LOUISVILLE L. REV. 333 (2021).

77. Impression Prods., Inc. v. Lexmark Int'l, Inc., 137 S. Ct. 1523, 1531-32 (2017). *See* Robert W. Gomulkiewicz, *Is the License Still the Product?*, 60 ARIZ. L. REV. 425 (2018) (comparing copyright's statutory approach to patent's common law approach to the exhaustion doctrine).

78. This would also touch on trade secret rights at both the state and federal levels as discussed, *supra*, in Section III.

about utilizing open source,[79] with some opting to privilege procurement of open source software at least in part because it provides the right to repair. The issues are nuanced and evolving as large companies such as Alphabet (Google), Amazon, Apple, IBM, Intel, Meta Platforms (Facebook), and Microsoft incorporate open source into their business models. Many companies now use a combination of binary use and open source code in their operations, leading some to observe that "mixed source" may be the best approach.[80]

All this suggests that legislative action may be premature or even unnecessary. Perhaps it is best to let the software industry's approach to repair evolve and mature, particularly because the government has a good seat at the table through its procurement power.[81] Two issues illustrate the prudence of legislative caution.

First, software repairs raise the question of who should be liable if damage or injury occurs due to the repair. This is a core issue in software transactions[82] and has particular resonance because of the cascading effect that is created when fixing one bug leads to other (sometimes more problematic) bugs.[83] The open source community has well settled and deeply held views on that issue. The open source community believes that anyone who contributes to open source development should not bear liability for those contributions. As the author of the Open Source Definition, Bruce Perens, puts it: "If free software authors lose their right to disclaim all warranties and find themselves getting sued over the performance of the programs that they've written, they'll stop contributing free software to the world. It's to our advantage as users to help

79. *See, e.g.*, EUROPEAN COMMISSION, ECODESIGN PREPARATORY STUDY ON MOBILE PHONES, SMART PHONES AND TABLETS: FINAL REPORT 419-20 (Feb. 2021).

80. *See* Greg R. Vetter, *Commercial Free and Open Source Software: Knowledge Production, Hybrid Appropriability, and Patents*, 77 FORDHAM L. REV. 2087 (2009). *See generally* Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?,* 20 HARV. J.L. & TECH. 1 (2006); Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265 (2004).

81. *See* THE PEOPLE'S CODE, (Aug. 8, 2016), https://obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code ("As agencies across the Federal Government take steps to improve access to their source code, the amount of available Federal open source software will grow. In the coming months, we will launch a new website – Code.gov – so that our nation can continue to unlock the tremendous potential of the Federal Government's software."); U.S. Dept. of Com., OPEN SOURCE CODE, https://www.commerce.gov/about/policies/source-code; U.S. Dept. of Def., DOD OPEN SOURCE SOFTWARE FAQ, (Oct. 28, 2021), https://dodcio.defense.gov/Open-Source-Software-FAQ/. State governments are also active procuring open source software. *See, e.g.*, *Code California Playbook*, CAL. DEPT. OF TECH. LETTERS, TL 18-02, https://codecagov-playbook.readthedocs.io/en/latest/policy/.

82. *See generally* GOMULKIEWICZ ET AL., *supra* note 44.

83. *See generally* MYERS ET AL., *supra* note 3, at 5-18.

the author protect this right."[84] That stance is instantiated in all open source licenses, including the GPL and the BSD License.

Second, the open source community holds distinct views on whether sharing (i.e., making available[85]) modifications should be voluntary or mandatory. Indeed, some of the most fervent debates by hackers[86] involve a debate about this proposition because it raises the issue of software "freedom." On one side of the freedom debate, the developers who follow the approach of Richard Stallman and the Free Software Foundation believe that developers should be *required* to share modifications because that will lead to the most code being available for free use.[87] In other words, the more free code made available, the more freedom. This mandatory "share alike" is enforced through the terms and conditions of the GPL and other so-called copyleft licenses.[88] On the other side of the debate, some developers believe that true freedom means the right to choose whether or not to share a modification. They agree that sharing code is often a good thing, but they think it is wise and fair to let developers pick and choose when to do so. Developers who follow this philosophy use so-called "permissive licenses" such as the BSD License.[89]

The main point is that the software industry has well vetted views on liability for repairs and on the distribution of repairs. Thus, lawmakers should tread carefully before legislating in this arena.[90] As suggested in the next section, it may be prudent to reframe and even rename what a "right to repair" should mean in the context of software repairs.

---

84. Bruce Perens, *The Open Source Definition*, *in* OPEN SOURCES, *supra* note 65, at 171, 181.

85. For software repair, "making available repair information and tools" (as urged by right to repair proponents) would usually mean providing access and rights to use source code.

86. The dual meaning of the word "hacker" provides a useful reminder that tinkering with software can be done for both constructive and destructive purposes. In common parlance "hacker" refers to programmers who create or use software for malicious purposes. *See* ERIC S. RAYMOND, THE NEW HACKER'S DICTIONARY 233-34 (3d ed. 19996). However, many in the software industry use the term "hacker" to refer to serious programmers who enjoy tinkering with code, as in "I'm hacking some code to fix that bug." *Id.* at 231. *See also* STEVEN LEVY, HACKERS: HEROES OF THE COMPUTER REVOLUTION (1984).

87. *See* RICHARD M. STALLMAN, *What is Copyleft? in* FREE SOFTWARE, FREE SOCIETY: THE SELECTED ESSAYS OF RICHARD M. STALLMAN 89 (Joshua Gay ed. 2002).

88. *See* Richard A. Stallman, *The GNU Operating System and the Free Software Movement*, *in* OPEN SOURCES, *supra* note 65, at 53, 59-60.

89. *See* HEATHER J. MEEKER, THE OPEN SOURCE ALTERNATIVE: UNDERSTANDING RISKS AND LEVERAGING OPPORTUNITIES (2008); LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW (2005).

90. With that said, however, Europe has already passed legislation addressing software repair. *See* Directive 2009/24/EC of the European Parliament and Council on the protection of computer programs, arts. 4(1), 5(1), 2009 O.J. (L 111).

C. RE-FRAMING AND RE-NAMING A "RIGHT TO REPAIR" SOFTWARE?

As mentioned above, consideration of a right to repair software has already been subsumed in the ongoing consideration of open source software. Over the past two decades in particular, the discussion of the pros and cons of open source has been robust and nuanced. However, it might also be fair and more useful to reframe and rename a "right to repair" software as a right to: (1) *revert* to prior versions of a software product; (2) *refuse* updates; and (3) *receive* repairs for a certain period of time.[91] In the context of software repairs, these "three R's" make the most sense.[92]

(i) Reverting—Software programmers have an insatiable appetite for updating their code. But not every up*date* is an up*grade* from the user's point of view. Instead, some updates actually degrade the useability of the software, at least for some users. When that is the case, some users want the ability to revert back to a prior version of the software. This ability to revert would include both access to and the right to use the prior version as well as any information necessary to restore the user's system to the prior condition and could address restoring support for applications and hardware devices that interface with the software.

(ii) Refusing—As mentioned earlier, many software companies now automatically install bug fixes, security patches, and other updates via the internet. On the one hand this is convenient for software users. On the other hand, customers do not welcome every new update or find the timing of the update disruptive. An ability to refuse updates could address both these concerns.

(iii) *Receiving*—Software programmers' drive to improve code also presents a business opportunity: the ability to sell updates. Software businesses give

---

91. Indeed, these "rights" are often the focus of policymakers in Europe. *See, e.g.*, EUROPEAN COMMISSION, EXPLANATORY MEMORANDUM FOR THE ECODESIGN CONSULTATION FORUM: ECODESIGN AND ENERGY LABELLING—MOBILE PHONES, CORDLESS PHONES AND TABLETS 8 (2021) ("Software updates of the operating system shall be provided for 5 years, comprising security updates and for at least the first 3 years also functionality updates; Such updates shall be provided within a reasonable time after the market introduction of a related release; Updates shall not have an adverse effect on device performance, or the user has to have the option to downgrade to the prior version of the operating system . . ."); EUROPEAN COMMISSION, PREPARATORY STUDY FOR THE ECODESIGN AND ENERGY LABELLING WORKING PLAN 2020-2024, at 8-29 (Feb. 2021); EUROPEAN COMMISSION, PREPARATORY STUDY ON MOBILE PHONES, SMART PHONES, AND TABLETS: FINAL REPORT 418-20 (Feb. 2021).

92. Related to repair, many software power users and tinkerers appreciate the ability to customize their software in various ways. *See* HALVORSON, *supra* note 21, at 169-83. Customization could be included in the repair zone when considering a right to repair software.

away some updates for free and charge for others, especially major upgrades to a product. From the consumer's point of view, the decision to pay for an upgrade is basically a decision to buy a new product. Consumers vary in their appetite for investing in upgrades, with some always upgrading to the "new and improved" and others sticking with the "tried and true." For consumers in the latter category, support for the prior version is critical. But, of course, software businesses have every incentive to move on from old versions—it is expensive and often increasingly complicated to continue to repair old versions. For this reason, software companies want to sunset repairs, but consumers want the software company to continue to make and provide repairs.

Should legislators intervene by mandating the three R's: a right to revert to a prior version, refuse updates, and/or to receive repairs for a certain period of time? Is this a matter of consumer protection? Fair competition? Good environmental stewardship? Or would such a mandate be unwarranted government intervention in normal business practices best left to market forces[93] because business models provide a variety of value propositions that depend on ongoing maintenance obligations and rights to new versions?[94] Whatever the answers to these questions, a better framing (and naming) of the issue than calling it a "right to repair" should enable crisper legislative decision making.

One idea would be for legislators or regulators to mandate or to nudge software companies to disclose what the consumer will receive by way of reversion rights as well as any rights to receive and refuse repairs. For example, this could be a separate section in the EULA that specifically addresses software repairs. Many EULAs do address repair issues as reflected in Appendix I, which provides excerpts from a variety of software EULAs. However, the content and clarity of this information could be improved

93. *See* Sarah E. Needleman, *Sony Videogame Unit Plans To Combine Two Services*, WALL ST. J., Mar. 30, 2022, at B4 (describing competition between Sony and Microsoft in video game subscription services). *Cf.* Ann-Marie Alcantara, *More Join Subscription Bandwagon*, WALL ST. J., Mar. 31, 2022, at B4 (describing how a variety of industries, including the restaurant and airlines industries, are experimenting with subscription business models).

94. Microsoft's EULA for Windows Server, for instance, provides the right to downgrade to a prior version. *See also* Nicole Nguyen, *That New Chromebook Has an Expiration Date*, WALL ST. J., Mar. 8, 2022, at A16 (describing the implications of Google's Auto Update Policy, which guarantees software updates and security support for a certain number of years, on the useful life of Chromebook computers); Brian X. Chen, *supra* note 12, at A12-13.

dramatically.[95] Improving disclosure is important because it supports consumer choice and could also improve competition on license terms.[96]

VI. CONCLUSION

Software is pervasive and plays an increasingly large role in the value of consumer products, so should a right to repair include software repair? A right to repair software must take into account the complexity of software repairs, especially as more software is embedded in goods or accessed from the cloud. Fortunately, the software industry has a long history of providing software repairs—bug fixes, security patches, and updates—through a variety of channels. Of particular note is the software industry's embrace of open source software which facilitates the ability of software developers to find and fix bugs and software subscriptions which include software repairs. As policymakers consider whether government intervention is warranted, the current landscape of software repair provides a useful point of departure, perhaps narrowing the focus to whether legislative or regulatory intervention would be warranted for providing the right to revert to a prior version of a software program, to refuse updates, or obligating a software developer to provide bug fixes and security patches for a certain period of time.

95. *See generally* Robert W. Gomulkiewicz, *Getting Serious About User-Friendly Mass Market Licensing for Software*, 12 GEO. MASON L. REV. 687 (2004).

96. *Cf.* Emily G. Brown, Comment, *Time to Pull the Plug? Empowering Consumers to Make End-of-Life Decisions for Electronic Devices Through Eco-Labels and Right to Repair*, 2020 U. ILL. J.L. TECH. & POLC'Y 227, 249-50 (2020); Rita Imran, *EU votes for Right to Repair law, France to label products with repairability ratings from 2021*, iTHINKDIFFERENT, (Nov. 27, 2020), https://www.ithinkdiff.com/eu-right-to-repair-france-repairability-ratings/.

APPENDIX: Updates and Reversions in EULA

Software Product	"Updates"	"Reversion"
Microsoft Windows Server	The software periodically checks for system updates and may install them for you. You may obtain updates only from Microsoft or authorized sources, and Microsoft may need to update your system to provide you with those updates. By accepting this agreement, you agree to receive these types of automatic updates without any additional notice.	**Downgrade Rights.** Instead of creating, storing, and using the software, for each permitted instance, you may create, store, and use an earlier version of the software for so long as Microsoft provides support for that earlier version as set forth in (aka.ms/windowslifecycle). This agreement applies to your use of the earlier versions. For the avoidance of doubt, by electing this downgrade option: (i) you will not have the right to create, store, or use a greater number of instances of the software than are permitted under this agreement, and (ii) you will need to acquire licenses for all cores in the physical server in accordance with Section 3 of this agreement. If the earlier version includes different components not covered in this agreement, the terms that are associated with those components in the earlier version of these editions apply to your use of them. Neither the manufacturer or installer, nor Microsoft is obligated to supply earlier versions or other editions to you. At

		any time, you may replace an earlier version or edition with this version and edition of the software.
Adobe	Updating. The Software may cause Customer's Computer, without additional notice, to automatically connect to the Internet (intermittently or on a regular basis) to (a) check for Updates that are available for download to and installation on the Computer and (b) notify Adobe of the results of installation attempts.	Subject to the Permitted Number of Computers for the Subscription Edition, Adobe may allow Customer to install and use the most recent prior version of the Subscription Edition and the current version of the Subscription Edition on the same Computer during the License Term. If the Software is an Update to a prior version of Adobe software (the "Prior Version"), then Customer's use of this Update is conditional upon its retention of the Prior Version. Therefore, if Customer validly transfers this Update pursuant to Section 4.6, the Customer must transfer the Prior Version along with it. If Customer wishes to use this Update in addition to the Prior Version, then Customer may only do so on the same Computer on which it has installed and is using the Prior Version. Any obligations that Adobe may have to support Prior Versions during the License Term may end upon the availability of this Update. No other use of the Update is permitted.

		Additional Updates may be licensed to Customer by Adobe with additional or different terms.
Corel Draw (General)	You acknowledge that We have no express or implied obligation to announce or make available any updates, enhancements, modifications, revisions, or additions to the Software and that this EULA does not give You any rights in or to any of the foregoing. We may also offer additional support and/or maintenance services for certain Software under the terms of a separate agreement. If You purchase such support and/or maintenance services with the Software, such services will be provided to You pursuant to the terms and conditions of that separate agreement. We reserve the right to amend, modify, suspend, or terminate Our support and/or maintenance policies at any time. We may, from time to time, download and install Software updates, bug fixes, feature enhancements, or improvements ("**Updates**") automatically on the devices under Your control or possession unless You decline such Updates beforehand. If You do not want to receive Updates, You must notify Us of Your choice, and, where the Software permits, disable the function that allows for automatic Updates. Otherwise, You agree to receive such Updates from Us as part of Your use of the Software. If the Update is not installed, You may not receive full benefit of the Software or	If You are an Entity that is acquiring, or allowing the acquiring of Perpetual Licenses or OEM Licenses for use of the Software on or through Your Assets, and want to use, or are allowing the use of, different (downgraded or upgraded) versions of such Software on or through Your Assets, then You must purchase: (a) a Subscription License for the business edition of the Software, where available, under the terms of the BULA, and/or (b) a Perpetual License for the business edition of the Software, where available, under the terms of the BULA along with maintenance and support services for such Software in accordance with Our terms for providing such services. Without such purchase You cannot use, and must not allow the use of, any different versions of the Software (other than the version associated with the Perpetual License and/or OEM License) on or through any of Your Assets.

	the Software may not perform properly. We have no obligation to provide any support to the Software without the installation of such Updates. We also have no obligation to create Updates on any schedule and retain sole discretion to make Updates available. If an Update is necessary to comply with applicable law, to address a threatened or actual security breach in the Software under license, to replace technologies that may infringe third-party intellectual property rights, or for any other reason of similar significance to Us ("**Mandatory Updates**"), We will deliver such Mandatory Update to You along with a notice that the Update is a Mandatory Update. You shall promptly install the Mandatory Updates, but in any event no later than ten (10) business days after receipt. Your failure to timely install Mandatory Updates may result in the termination or suspension of Your license(s) for affected Software. We may also, from time to time, perform scheduled maintenance of the infrastructure and programming used to provide the Software, during which time You may experience some disruption to that Software or access to any associated accounts or services. Whenever reasonably practicable, We will provide You with advance notice of such maintenance. You acknowledge that, from time to time, We may need to perform emergency maintenance without providing You advance notice, during which time We may temporarily suspend Your access to, and use of, the Software or any	

	associated accounts or services.	
MindManager (App under Corel Draw)	A Subscription License to the Software entitles You to receive free Product Upgrades and Product Updates. A Perpetual License to the Software entitles You to receive Patches free of charge, for the first twelve months of the Perpetual Term. A Perpetual License does not entitle You to receive any Product Upgrades or Product Updates free of charge. Except as otherwise provided at the time of download or provision by Us, any supplemental software code or related materials that We provide to You as part of any support services, paid or otherwise, are to be considered part of the Software and are subject to this EULA. We may use any technical information You provide to Us for any business purposes, without restriction, including for product support and development. ---Definitions--- Product Upgrades: Product Upgrades, also known as Major Releases, introduce important new features or significantly enhance existing Product functionality. These releases are made available on a product-by-product basis. Product Upgrades are available at no additional charge to purchasers of MindManager Software Assurance and Support (MSA), Upgrade Protection Plan or available without the purchase of MSA or Upgrade Protection Plan for an additional charge. Product Updates: Product Updates, also known as Minor or Maintenance	Any upgrade You accept to receive from Us and install, run or use in respect of an earlier version of the Software shall: (i) automatically cancel and terminate Your prior agreement through which You obtained a license for the earlier version of the Software from us, and (ii) cause this EULA to replace and supersede such prior agreement for the Software version You upgraded from. Upon such upgrade, You may no longer use the earlier version of the Software unless otherwise specified in a License Certificate or any other services agreement entered into between Us and You for the Software. We reserve the right to require certification of the destruction and removal of such previous version.

	Releases, incorporate service packs that provide bug fixes and, may also include, other minor fixes or modifications that enhance Product usage or functionality. These releases are made available on a product-by-product basis. Product Updates are provided free of charge and will be made available for download. Patches: If MindManager discovers a significant problem after a Product has shipped, MindManager produces patches on an as-needed basis to provide interim or emergency fixes to one or more critical problems. Potentially affected customers are alerted with instructions on how to obtain and apply the necessary patch or run the registry script executable.	
AutoCAD	During the period of Your subscription, Autodesk may make available or deliver Updates or Upgrades to Software. All such Updates and Upgrades are subject to the same license and other terms as the Software to which the Updates or Upgrades apply. You are encouraged to promptly install and use all Updates and Upgrades made available to You during the subscription period. ---Definitions--- **Software** means any software or similar materials, including any modules, components, features and functions, made available by Autodesk, whether or not provided as part of a subscription and whether or not provided for a fee. Software includes Updates and Upgrades. **Updates** means security fixes, hot fixes, patches and other updates	If You receive an Update or Upgrade for any Software, You may install and use both the previous version and the new version of the Software for testing and migration purposes for a maximum of 120 days (beginning on the first installation date for the new version), provided that, during such 120-day period, You do not use both versions concurrently for production use. After such 120 days, (i) Your (including Your Authorized Users') right to access and use such previous version will end, and (ii) You must stop all access to and use of the previous version (including all access and use by Your Authorized Users),

	(including new features, new functions and other modifications released between Upgrades), if and when made available to You by Autodesk and determined by Autodesk to constitute an update. **Upgrades** means new versions of Offerings, or add-ons to or additional products associated with Offerings, if and when made available to You by Autodesk and determined by Autodesk to constitute an upgrade.	uninstall all copies of the previous version, and, at Autodesk's request, destroy any such copies or return them to Autodesk or the reseller from which You acquired the Offering. For certain Offerings (because of Special Terms for the Offerings or because of exceptions granted by Autodesk under certain circumstances), You may have certain rights to continue using and accessing previous versions after such 120-day period. Such rights, if any, are set forth in the Previous Version Rights (see Subscription Benefits). For the duration of a subscription, You may make one archival copy of the Software to which You subscribed solely for Your backup and archival purposes.
APPLE iOS15/ iPadOS 15	The software (including Boot ROM code, embedded software and third party software), documentation, interfaces, content, fonts and any data that came with your Device ("Original Apple Software"), as may be updated or replaced by feature enhancements, software updates or system restore software provided by Apple ("Apple Software Updates"), whether in read only memory, on any other media or in any other form (the Original Apple Software and Apple Software Updates are collectively referred to as the "Apple Software")	N/A

	are licensed, not sold, to you by Apple Inc. ("Apple") for use only under the terms of this License Apple, at its discretion, may make available future Apple Software Updates. The Apple Software Updates, if any, may not necessarily include all existing software features or new features that Apple releases for newer or other models of Devices. The terms of this License will govern any Apple Software Updates provided by Apple, unless such Apple Software Update is accompanied by a separate license, in which case you agree that the terms of that license will govern. Your Device will periodically check with Apple for Apple Software Updates. If an update is available, the update may automatically download and install onto your Device and, if applicable, your peripheral devices. By using the Apple Software, you agree that Apple may download and install automatic Apple Software Updates onto your Device and your peripheral devices. You can turn off automatic updates altogether at any time by changing the Automatic Updates settings found within Settings > General > Software Update. Subject to the terms and conditions of this License, you are granted a limited non-exclusive license to download Apple Software Updates that may be made available by Apple for your model of the Device to update or restore the software on any such Device that you own or control.	

	This License does not allow you to update or restore any Device that you do not control or own, and you may not distribute or make the Apple Software Updates available over a network where they could be used by multiple Devices or multiple computers at the same time. If you download an Apple Software Update to your computer, you may make one copy of the Apple Software Updates stored on your computer in machine-readable form for backup purposes only, provided that the backup copy must include all copyright or other proprietary notices contained on the original. If you choose to allow automatic app updates, your Device will periodically check with Apple for updates to the apps on your Device and, if one is available, the update will automatically download and install onto your Device. You can turn off the automatic app updates altogether at any time by going to Settings, tap iTunes & App Store, and under Automatic Downloads, turn off Updates.	
MacOS (Monterey)	Apple, at its discretion, may make available future upgrades or updates to the Apple Software for your Apple-branded computer. Upgrades and updates, if any, may not necessarily include all existing software features or new features that Apple releases for newer or other models of Apple-branded computers. The terms of this License will govern any software upgrades or updates provided by Apple that replace and/or supplement the original Apple Software product,	N/A

	unless such upgrade or update is accompanied by a separate license in which case the terms of that license will govern. Apple is not obligated to provide any updates, maintenance, warranty, technical or other support, or services for the resultant modified Apple Software. You expressly acknowledge that if failure or damage to Apple hardware results from modification of the Open-Sourced Components of the Apple Software, such failure or damage is excluded from the terms of the Apple hardware warranty. Apple has provided as part of the Apple Software package, and may provide as an upgrade, update or supplement to the Apple Software, access to certain third party software or services as a convenience. To the extent that the Apple Software contains or provides access to any third party software or services, Apple has no express or implied obligation to provide any technical or other support for such software or services. The Apple Software will periodically check with Apple for updates to the Apple Software. If an update is available, the update may automatically download and install onto your computer and, if applicable, your peripheral devices. By using the Apple Software, you agree that Apple may download and install automatic updates onto your computer and your peripheral devices. You can turn off automatic	

	updates altogether at any time by changing the automatic updates settings found within System Preferences.	
Catalyst Production Suite (Sony)	The Software includes all of the software in your Product (defined below), including updates or modified software provided to you by Sony, whether stored on media or downloaded via any method, but not Excluded Software as defined below. From time to time, Sony may automatically update or otherwise modify the Software, including, but not limited to, for purposes of enhancement of security functions, error correction and improvement of functions, at such time as you interact with Sony's or third parties' servers, or otherwise. Such updates or modifications may delete or change the nature of features or other aspects of the Software, including, but not limited to, functions you may rely upon. You acknowledge and agree that such activities may occur at Sony's sole discretion and that Sony may condition continued use of the Software upon your complete installation or acceptance of such update or modifications. Any updates/modifications shall be deemed to be, and shall constitute part of, the Software for purposes of this EULA. By acceptance of this EULA, you consent to such update/ modification.	N/A
Riot Games	In an effort to constantly improve the Riot Services, evolve our games and keep the Riot Services, safe, fun,	N/A

	and secure, you agree that we may change, modify, update, suspend, "nerf," or restrict your access to any features or parts of the Riot Services, including Virtual Goods (e.g., we might change some features of Virtual Goods for regulatory or legal reasons or to improve the game experience), and may require that you download and install software and updates to any software required to support the Riot Services, at any time without liability to you. You also understand and agree that any such changes or updates to the Riot Services might change the system specifications necessary to play our games, and in such a case, you, and not Riot Games, are responsible for purchasing any necessary additional software or hardware in order to access and play our games. You also understand and agree that we may use background patching to automatically update our games and software with or without notice to you.	
Epic Games	Epic may provide patches, updates, or upgrades to the Software that must be installed in order for you to continue to use the Software or Services. Epic may update the Software remotely without notifying you, and you hereby consent to Epic applying patches, updates, and upgrades. Epic may modify, suspend, discontinue, substitute, replace, or limit your access to any aspect of the Software or Services at any time. You acknowledge that your use of the Software or Services does not confer on you any interest, monetary or otherwise, in any aspect or feature of the Software or Services, including	N/A

	but not limited to any in-game rewards, achievements, character levels. You also acknowledge that any character data, game progress, game customization or other data related to your use of the Software or Services may cease to be available to you at any time without notice from Epic, including without limitation after a patch, update, or upgrade is applied by Epic. Epic does not have any maintenance or support obligations with respect to the Software or Services. ---Definition--- The term "Software" also includes any patches, updates, and upgrades to such Software, and all related content and documentation provided with or for the Software, additionally including but not limited to all software code, titles, themes, objects, characters, names, dialogue, catch phrases, locations, stories, artwork, animation, concepts, sounds, audio-visual effects, methods of operation, and musical compositions that are related to such Software, and any copies of any of the foregoing.	
Evernote	In connection with any modification of the Evernote Service, Evernote may automatically download software updates on your computers and devices from time to time with the intention of improving, enhancing, repairing and/or further developing the Evernote Service. Evernote will endeavor to provide you with the option of whether or not to install the update; however, in certain circumstances (e.g., security risks), Evernote may require you to	N/A

	install the update to continue accessing the Evernote Service. In all cases, you agree to permit Evernote to deliver these updates to you (and you to receive them) as part of your use of the Evernote Service.	
Valve Steam	As a Subscriber you may obtain access to certain services, software and content available to Subscribers or purchase certain Hardware (as defined below) on Steam. The Steam client software and any other software, content, and updates you download or access via Steam, including but not limited to Valve or third-party video games and in-game content, software associated with Hardware and any virtual items you trade, sell or purchase in a Steam Subscription Marketplace are referred to in this Agreement as "Content and Services;" the rights to access and/or use any Content and Services accessible through Steam are referred to in this Agreement as "Subscriptions." For reasons that include, without limitation, system security, stability, and multiplayer interoperability, Valve may need to automatically update, pre-load, create new versions of or otherwise enhance the Content and Services and accordingly, the system requirements to use the Content and Services may change over time. You consent to such automatic updating. You understand that this Agreement (including applicable Subscription Terms) does not entitle you to future updates (unless to the extent required by applicable law), new versions or other enhancements	N/A

	of the Content and Services associated with a particular Subscription, although Valve may choose to provide such updates, etc. in its sole discretion.	
Final Fantasy VII Remastered	For the purposes of this EULA, references to the Software Product includes computer software owned by SEL or its third party suppliers/ licensors and associated media, any printed materials, manuals, any on-line or other documentation together with, to the extent not distributed with a separate licence agreement, any updates or patches to the original game software which are provided to you or which you may download from any SEL web site or other source authorised by SEL expressly for such purpose) including such software required in order to access and/or use any on-line features and functionality which may be associated with such computer game software (subject to any additional terms of use applicable to such on-line mode). You understand that the Software Product may be updated or patched at any time and in doing so no obligation to provide such updates or patches to you pursuant to this EULA or otherwise shall arise. Your installations and use of any updates or modifications to the Software Product or your continued use of the Product Software following notice of changes to this EULA will constitute your acceptance of any and all such	N/A

	changes to the terms of this EULA. This EULA is intended to govern your use of the Wrapper Software *(third party plugin)* and any updates or patches to it, which are provided to you or which you may download from any SEL website, or other source authorised by SEL, expressly for such purpose.	
CloudLinux OS	For purposes of this Agreement, the "Programs" include any updates, enhancements, modifications, revisions, or additions to the Programs made by Cloud Linux and made available to end-users. Notwithstanding the foregoing, Cloud Linux shall be under no obligation to provide any updates, enhancements, modifications, revisions, or additions to the Programs.	N/A
Oracle (Software Technical Support)	Lifetime Support consists of the following service levels: Premier Support (also referred to as, and will be documented on your order as, "Software Update License & Support" or "Oracle Communications Network Software Premier Support") Extended Support (if offered) Sustaining Support **Program Updates** Update means a subsequent release of the program which Oracle generally makes available for program licenses to its supported customers at no additional license fee, other than shipping charges if applicable, provided you have ordered a technical support offering that includes software updates for	N/A

	such licenses for the relevant time period. Updates do not include any release, option or future program that Oracle licenses separately. Updates are provided when available (as determined by Oracle) and may not include all versions previously available for a program acquired by Oracle. Oracle is under no obligation to develop any future programs or functionality. Any updates made available will be delivered to you, or made available to you for download. If delivered, you will receive one update copy for each supported operating system for which your program licenses were ordered. You shall be responsible for copying, downloading and installing the updates. **Software Update License & Support** Program releases in the Premier Support phase of Oracle's product support lifecycle will receive Software Update License & Support. Software Update License & Support consists of: Program updates, fixes, security alerts and critical patch updates Upgrade scripts (availability may vary by program)	
TeamViewer	The Software Specific Terms contain the terms and conditions that additionally apply to the use of: (i) software provided by TeamViewer, whether installed on devices of the Customer or accessed via web browser, also including any applications (e.g., apps for mobile terminals), add-on components, customized settings and features, and	N/A

	all updates and Release Versions as herein below defined thereof (collectively "**Software**"), and (ii) servers for the establishment of encrypted connections (handshake) and for the forwarding of data packets (routing) in connection with the use of the Software ("**Server Services**"), as well as (iii) any further cloud-based services provided by TeamViewer. TeamViewer reserves the right to change the Software in the context of updates and/or Release Versions as well as the other Services (including the System Requirements) for good cause. Such good cause exists especially if the change is required due to (i) a necessary adaptation required by applicable law, regulation, court order, or order of authority; (ii) changes to applicable technical framework conditions (e.g., new encryption standards); or (iii) the protection of system security. TeamViewer may, at its sole discretion, but shall not be obligated to, provide releases of the Software for download ("**Release Versions**"). Additional features to the Software which are separately marketed and/or priced by TeamViewer ("**Additional Features**") shall not qualify as Release Versions. All rights of use set forth in the Contract applicable to the Software shall also apply to Release Versions. Customer is obliged to update the Software with any Release Version at its own cost as soon as reasonably	

	practicable. Customer's systems shall comply with the System Requirements to accommodate new Release Versions. Any malfunctioning of the Software or failure in the Services that is attributable to non-compliance with this section shall be Customer's sole responsibility. The obligation of Customer holding a previously acquired perpetual license to update the Software shall be limited to the minor Release Versions (e.g., version XX.1, XX.2 "**Minor Release Version**") relating to the main version (e.g., version XX, YY) for which the Perpetual License was acquired. Minor Release Versions may contain the correction of errors, security patches as well as minor improvements of functions (e.g., optimizations in the program execution speed) and will be marked by TeamViewer – in its sole discretion – by a change in the number behind the main version number. TeamViewer shall use commercially reasonable efforts to eliminate Errors within a reasonable period of time following Customer's notification of such Errors, for which Customer shall provide comprehensive details of the circumstances relating to the Errors and supporting documentation (e.g., screenshots, protocol data) in its notification, as far as this is possible and can be reasonably expected. TeamViewer may, at its sole option, eliminate Errors by delivering patches or updates, through Release Versions or otherwise. If the elimination of an Error is not	

	available using financially reasonable efforts within a predictable time, TeamViewer shall be entitled to provide temporary workarounds for such Error, provided that the functionalities and availability of the Services are not materially affected. Unless otherwise agreed to by the Parties, the Professional Services in relation to the Software (e. g. installation, configuration, application, integration, update), will be provided for the most current version of the Software. Customer is committed to the fulfilment and maintenance of the System Requirements as set out in the EULA for the respective Software. In case that the installation and/or update of the Software is not part of the Professional Services, Customer shall ensure that it has the respective Software installed and updated to the then-current version on its computers (desktop PC or notebook) or mobile devices (e.g., iOS, Android) for the duration of the Professional Services.	
Intuit Quickbooks	You will manage your passwords and accept updates. You are responsible for securely managing your password(s) for access to the Software and to contact Intuit If you become aware of any unauthorized access to your account. The Software may periodically be updated with tools, utilities, improvements, third party applications, or general updates to improve the Software. You agree to receive and install these updates. If you obtained a subscription for the	N/A

	Software that includes new or Upgrade versions (as defined below) of the Software, you agree that Intuit may require you to install such new or Upgrade versions of the Software in order to continue your subscription. You agree to accept and install all such new or Upgrade version(s) of the Software within the time period specified by Intuit. You understand and agree that if you do not install such new or Upgrade version(s) within the specified time period Intuit may provide you a second and final notification that accepting such new or Upgrade version(s) is required and that failure to install such new or Upgrade version and replace the prior version of the Software will result in termination of your subscription. If you do not then make such Upgrade within an additional specified time period after the date of issuance of Intuit's second and final notification then Intuit, at its sole discretion, may immediately terminate (i) your subscription, (ii) your continued use of the Software and (iii) all other subscription benefits and services; and, at its discretion, refund any unused or prorated balance of your subscription fees. ---Definition--- "Enhancement(s)" means any and all minor enrichments to the Software, such as new or improved features, functionality, compatibility, performance, or other content or information. For clarity, Enhancements exclude Updates and Upgrades.	

	"Software" has the meaning defined above in Section A, 1.1., and includes the QuickBooks Desktop software that is the object of this Agreement, any Intuit-provided Services, software, applications, programs, tools, and other components, accessible in or through, or in combination with, QuickBooks Desktop, including but not limited to the QuickBooks Desktop Manager installer application and the QuickBooks Desktop Mobile Application(s) for iOS and/or Android mobile operating systems, if available for use with your version of QuickBooks, as well as all Updates that you may be eligible to receive based on the license or Subscription purchased as set forth in Section 10 further below. For clarity, Software excludes Upgrades. "Updates" means Software bug fixes and error corrections generally provided to users of your specific version of the Software, when-and-if they are made available. For clarity, Updates exclude Enhancements and Upgrades. "Upgrades" means each and all major or significant future-released versions of the full or complete Software. For clarity, Upgrades exclude Enhancements and Updates.	
Cisco	Upgrades or Additional Copies of Software. You may only use Upgrades or additional copies of the Software beyond Your license Entitlement if You have (a) acquired such rights under a support agreement covering the applicable Software; or (b) You have purchased	N/A

	the right to use Upgrades or additional copies separately. ---Definition--- "Software" means the Cisco computer programs including Upgrades, firmware and applicable Documentation. "Upgrades" means all updates, upgrades, bug fixes, error corrections, enhancements and other modifications to the Software.	
VMware (General)	**SUPPORT SERVICES.** Support and subscription services for the Software ("Support Services") are provided pursuant to the Support Services Terms and are not subject to this EULA. You have no rights to any updates, upgrades or extensions or enhancements to the Software unless you separately purchase Support Services or they are included with your purchase of a license to the Software as provided in the Product Guide.	N/A
VMware (Technical Support and Subscription)	**"Subscription Services"** means any Maintenance Releases, Minor Releases, and Major Releases to the Software and related Documentation that VMware provides to Customer. (a) **"Maintenance Release"** or **"Update"** means a generally available release of the Software that typically provides maintenance corrections only or high severity bug fixes, designated by means of a change in the digit to the right of the second decimal point (e.g., Software 5.0 >> Software 5.0.1), or for certain Software by a change in the digit of the Update number (e.g., Software 5.0 Update 1).	N/A

	(b) **"Minor Release"** means a generally available release of the Software that: (i) introduces a limited number of new features, functionality, and minor enhancements; (ii) fixes for high severity and high priority bugs identified in the current release; and (iii) is designated by a change in the digit to the right of the decimal point (e.g., Software 5.0>>Software 5.1). (c) **"Major Release"** also known as an "Upgrade" means a generally available release of the Software that: (i) contains functional enhancements and extensions; (ii) fixes for high severity and high priority bugs; and (iii) is designated by VMware by a change in the digit to the left of the first decimal point (e.g., Software 5.0 >> Software 6.0). VMware may, at its discretion, offer complimentary Services, including for certain Software, as more fully described on the VMware Technical Support Services website, at https://www.vmware.com/support/services/complimentary.html. **"VMware Complimentary Update Services"** means the provision of Maintenance Releases and Minor Releases to Customer, at no cost. This VMware Complimentary Update Service does not include providing any Major Releases.	

Samsara	**"Samsara Software"** means the Apps, Firmware, and Hosted Software, and any improvements, modifications, patches, updates, and upgrades thereto that Samsara develops or provides in connection with these Terms, and Support Services. General. Samsara continuously improves the Products, and may from time to time (i) update the Samsara Software and cause Firmware updates to be automatically installed onto Hardware; (ii) update the Apps; or (iii) upgrade Hardware equipment to newer models. Samsara may change or discontinue all or any part of the Products, at any time and without notice, at Samsara's sole discretion. If Samsara discontinues supporting the Products or Services you have ordered from Samsara in accordance with these Terms prior to the applicable License Expiration Date without offering to replace them with an updated version or newer model, you may request a Refund. Updates or upgrades may include security or bug fixes, performance enhancements, or new functionality, and may be issued with or without prior notification to Customer. Customer hereby consents to such automatic updates.	N/A