CRACKING THE CODE: HOW TO PREVENT COPYRIGHT TERMINATION FROM UPENDING THE PROPRIETARY AND OPEN SOURCE SOFTWARE MARKETS

Grant Emrich[*]

Computer software is protected by copyright law through its underlying code, which courts have interpreted as constituting a "literary work" pursuant to the Copyright Act. Prior to including software as copyrightable subject matter, Congress established a termination right which grants original authors the ability to reclaim their copyright thirty-five years after they have transferred it. Termination was intended to benefit up-and-coming authors who faced an inherent disadvantage in the market when selling the rights to their works. In the near future, many software works will reach the thirty-five-year threshold, thus presenting courts with a novel application of termination to computer software.

Software's inclusion as copyrightable subject matter has long been seen as a poor fit when compared to other copyrightable works, such as music, movies, and art. This perceived difference will soon be exacerbated because termination poses unique threats as applied to software, primarily due to the functional aspects of software that are necessarily incidental to the protected code.

Problems stemming from termination will manifest differently in the two primary software markets known as proprietary software and open source software. Independent contractors may be able to terminate copyrights held in software they had previously written for a business's proprietary ownership, whereas, in the context of open source software, exercise of termination could make void perpetual licensing agreements that serve as the foundation for the open source movement. While statutory and common law exceptions to termination, such as the work made for hire doctrine, may mitigate the effects of termination, the degree to which the doctrines may do so has yet to be determined.

[*] J.D. Candidate 2022, Fordham University School of Law; B.A., 2018, Binghamton University. I would first like to thank Professor Ron Lazebnik for his insight and guidance throughout the writing process. Additionally, thank you to the editors and staff of the *Fordham Law Review* for their careful edits and constructive comments. Finally, thank you to my wonderful family for their unconditional love and support. To my mother, Alecia, father, Michael, and brothers, Michael, Brendan, Garrett, and Dean, I couldn't have done this without you.

This Note argues that the harmful effects of termination as applied to proprietary software can be resolved through a novel interpretation of the work made for hire provision of the Copyright Act. Additionally, the harmful effects of termination on open source software can be avoided if Congress adopts a legislative amendment creating a compulsory licensing system for open source works.

INTRODUCTION

Just over twenty years ago, international panic spread as survival supplies were stockpiled,[1] planes were grounded,[2] and many prepared for what was feared to be the end of the world.[3] This event, most commonly referred to as "Y2K," was borne out of a combination of negligent coding practices and the absence of oversight.[4] Nearly all computers at the time were programmed to track the current year based on its last two digits, which became problematic when the imminent change to the year 2000 would appear indistinguishable from the year 1900 to computer systems.[5] Although the ramifications were speculative at the time, this malfunction had the potential to cause massive technological failures in all computer-reliant systems ranging from aviation to financial markets.[6] The potential catastrophe led to the creation of the Senate Special Committee on the Year 2000 Technology Problem, which oversaw the process of rewriting millions of lines of code.[7]

The dreaded disaster never came to pass, leading much of the general public to believe that no genuine threat had ever existed in the first place.[8] Although this sentiment may have been true regarding the most hyperbolic prognostications, hundreds of technological failures were still experienced as a result of the underlying bug.[9] Furthermore, the results that could have been experienced may never be known due to the preventative steps taken by countries around the world.[10]

Regardless of its outcome, the Y2K event exemplified the importance of the ability to both access and alter computer software code. An underexplored provision of copyright law, known as the "termination right," may introduce a substantial barrier to code accessibility in the coming years.

1. Kris Epley, *Residents Stockpile Supplies in Fear of Y2K Chaos*, GRAND ISLAND INDEP. (Dec. 16, 2011), https://theindependent.com/news/residents-stockpile-supplies-in-fear-of-y2k-chaos/article_1789cb1b-2e95-5a80-a36a-92fd93cf993e.html [https://perma.cc/JEQ4-EFY7].

2. *See* April F. Robbins, *Aviation and the Year 2000: What's the Big Deal?*, 64 J. AIR L. & COM. 835, 836 (1999).

3. *See* Lily Rothman, *Remember Y2K?: Here's How We Prepped for the Non-Disaster*, TIME (Dec. 31, 2014, 12:00 PM), https://time.com/3645828/y2k-look-back/ [https://perma.cc/4HQD-C82C].

4. Dylan Mulvin, *Distributing Liability: The Legal and Political Battles of Y2K*, 42 IEEE ANNALS OF THE HIST. OF COMPUTING, no. 3, 2020, at 26, 28 ("[T]he widely distributed antecedents to the Y2K crisis include economic imperatives, bureaucratic decisions, haphazard coding techniques, a lack of managerial oversight, and scant attention paid to software maintenance.").

5. *See* Zachary Loeb, *The Lessons of Y2K, 20 Years Later*, WASH. POST (Dec. 30, 2019), https://www.washingtonpost.com/outlook/2019/12/30/lessons-yk-years-later/ [https://perma.cc/Z7AX-DSDV].

6. *Id.*

7. *Id.*

8. *Id.*

9. *Id.* ("And while many believed that 'nothing happened,' there were actually hundreds of Y2K-related incidents. These included problems at more than a dozen nuclear power plants, delays in millions of dollars of Medicare payments, ATM issues worldwide and problems with the Defense Department's satellite-based intelligence system.").

10. *Id.*

While unlikely to precipitate the end of the world, the termination right does have the potential to significantly disrupt the computer software industry and cause issues throughout our modern tech-reliant economy.

The termination right is a powerful tool that allows original authors of copyrighted work to regain ownership rights thirty-five years from the date they transferred or licensed the rights to others.[11] Given the popularization of computer software in the 1980s and its subsequent classification as a "literary work" protectable under the Copyright Act of 1976[12] ("Copyright Act"), many software works will soon reach the thirty-five-year threshold, thus becoming eligible for copyright termination.[13]

The software industry has flourished in the years since courts began interpreting the Copyright Act to protect software as copyrightable subject matter.[14] Due to the judicial origins of software copyright protection, Congress likely did not foresee application of the termination right to software when enacting the Copyright Act.[15] Although Congress included limitations on termination in the Copyright Act, the limitations' effectiveness in mitigating the effects of software terminability are uncertain, as the issue has yet to reach the courts.[16]

Software terminability may have markedly different consequences for the two primary types of software: proprietary software and open source software (OSS).[17] For proprietary software, companies that hired independent contractors to create their software programs could unexpectedly have their rights in such programs stripped away when said contractors exercise their termination rights decades later.[18] This is especially problematic due to the interconnected nature of software development, which could lead to a chain effect wherein the termination of a software copyright could jeopardize rights held in current programs, while simultaneously prohibiting further use of the original code in developing new programs.[19]

11. 17 U.S.C. § 203(a)(3).

12. Pub. L. No. 94-553, 90 Stat. 2541 (codified as amended in scattered sections of the U.S.C.); *see* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1249 (3d Cir. 1983).

13. *See* Timothy K. Armstrong, *Shrinking the Commons: Termination of Copyright Licenses and Transfers for the Benefit of the Public*, 47 HARV. J. ON LEGIS. 359, 362 (2009) (explaining that a particular form of software license is too recent to fall within the termination window).

14. *See* Chris Hopfensperger, *Software: Growing US Jobs and the GDP*, SOFTWARE.ORG (Sept. 19, 2019), https://software.org/blog/software-growing-us-jobs-and-gdp/ [https://perma.cc/2CB4-N44S].

15. While legislative history indicates that Congress did intend to include software as copyrightable subject matter, Congress did not discuss termination. *See* H.R. REP. NO. 94-1476, at 53–54 (1976); *infra* note 91.

16. *See* Armstrong, *supra* note 13, at 422–23.

17. *See infra* Part I.E.

18. *See* Grant C. Yang, *The Continuing Debate of Software Patents and the Open Source Movement*, 13 TEX. INTELL. PROP. L.J. 171, 201–02 (2005).

19. Jon L. Phelps, *Copyleft Termination: Will the Termination Provision of the Copyright Act of 1976 Undermine the Free Software Foundation's General Public License?*, 50 JURIMETRICS 261, 266 (2010) ("[T]he terminated licensee is prohibited from creating further

Software terminability presents different complications for OSS, a unique type of software that is made available to the public for use, alteration, and overall improvement.[20] The OSS movement has thrived due to certain underlying precepts which run counter to the proprietary model.[21] Foremost among these precepts is that the open availability of code for others to access and alter in a communal fashion is instrumental toward the progression of software development.[22] The introduction of software termination may throw the current system into flux, with commentators warning that "copyright termination may be the Achilles' heel"[23] and a "potential chink in the armor"[24] of many open source programs. If OSS authors exercise termination rights, previously available code will be rescinded, thus stifling current use, prohibiting future development, and producing a chilling effect on programmers' future reliance on OSS.[25]

The impending thirty-five-year terminability threshold and the proliferation of proprietary software and OSS are on a collision course.[26] In anticipation of this conflict, this Note seeks to delineate the scope and severity of problems created by software terminability and to examine solutions that could help avoid this collision altogether. Part I of this Note provides the background necessary to understand the interactions between copyright law, termination, and computer software. Part II explores potential statutory and common law exemptions from termination, which will clarify the types of software most at risk of being terminated. Lastly, Part III proposes a statutory interpretation solution to mitigate termination's effect on proprietary software, as well as a legislative solution to insulate OSS from termination.

I. THE CONVERGENCE OF COPYRIGHT LAW, TERMINATION RIGHTS, AND THE SOFTWARE INDUSTRY

Copyright is a form of intellectual property protection granted to "original works of authorship fixed in any tangible medium of expression."[27] Copyright protection has existed in some form in the United States since the nation's founding.[28] Although neither software nor termination rights

'derivative works based upon the copyrighted work covered by the terminated grant.'" (quoting 17 U.S.C. § 203(b)(1))).

20. *See id.* at 263. Although often referred to as "Free and Open Source Software," this title does not necessarily mean that the software is free of charge, as licensees may still have to pay for their use. Rather, it is intended to allude to the various freedoms associated with open access. *See* VÍCTOR VÁZQUEZ LOPEZ, INTERNATIONAL IP PROTECTION OF SOFTWARE: HISTORY, PURPOSE AND CHALLENGES 8 (2007).

21. *See* Armstrong, *supra* note 13, at 361–62.

22. *See* Phelps, *supra* note 19, at 263.

23. Yang, *supra* note 18, at 201.

24. Phelps, *supra* note 19, at 269.

25. *Id.* at 271–72.

26. *See* Armstrong, *supra* note 13, at 422.

27. 17 U.S.C. § 102(a).

28. 2 PETER S. MENELL ET AL., INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE 2020: COPYRIGHTS, TRADEMARKS AND STATE IP PROTECTIONS 508 (2020).

existed at the time of copyright's inception, underlying theoretical justifications and policy rationales for copyright protection are still useful in elucidating issues stemming from software terminability. Furthermore, insight into the software industry's technical and economic intricacies helps to determine the scope of the impact that software terminability may eventually have.

Part I of this Note establishes the legal, technical, and economic backdrop against which the issues posed by software terminability become apparent. Part I.A explains modern copyright law, including its constitutional basis and statutory embodiment in the Copyright Act. Part I.B analyzes the legislative history and theoretical justifications underlying the termination right. Part I.C examines the work made for hire exception to terminability and its prospective applicability to software. Part I.D addresses software and its inclusion as copyrightable subject matter. Finally, Part I.E describes the current state of the software industry in both its proprietary and open source sectors.

A. The Development of Modern Copyright Law

The origins of copyright law in the United States can be traced back to the U.S. Constitution, which grants Congress the power to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[29] This clause contains an underlying tension between the interests of individual authors, who seek to prevent their creative works from being copied, and the public, which benefits from open accessibility to said creative works.[30] Although both interests must be balanced, the U.S. Supreme Court has interpreted the clause to prioritize the public interest and found that the protection of authors' rights is a necessary step in doing so.[31] This line of reasoning has been described as "utilitarian" and posits that the best way to ensure a robust body of publicly available works is to provide incentives to authors to create.[32] The federal government provides these incentives, which grant authors copyrights over their works.[33] The value of a copyright is derived from the exclusive rights it entails, as the potential enforcement of these rights against others grants authors, in essence, a limited monopoly over the exploitation of their works.[34]

Congress has exercised the power granted to it by the Constitution by enacting legislation that defines and protects copyright and its associated

29. U.S. CONST. art. I, § 8, cl. 8.

30. *See* MENELL ET AL., *supra* note 28, at 514 ("American copyright law can thus be seen as primarily striving to achieve an optimal balance between fostering incentives for the creation of literary and artistic works and the optimal use and dissemination of such works.").

31. *See* Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417, 429 (1984).

32. *See* Jeanne C. Fromer, *Expressive Incentives in Intellectual Property*, 98 VA. L. REV. 1745, 1751 (2012).

33. Edward C. Walterscheid, *To Promote the Progress of Science and Useful Arts: The Anatomy of a Congressional Power*, 43 IDEA 1, 20–21 (2003).

34. *Id.*

rights.[35] Although it has been amended multiple times, the most recent and currently controlling legislation is the Copyright Act.[36] The Copyright Act contains provisions detailing both the subject matter that is protectable under copyright[37] and the exclusive rights that are to be protected.[38] The Copyright Act also establishes the duration of copyright, which currently lasts for the life of the author, plus seventy years following the author's death.[39] Once a copyright expires, the work enters the "public domain," where it remains free and open for the public to use indefinitely.[40]

Whether a work constitutes copyrightable subject matter is a vital threshold matter because authors of works that do not qualify cannot obtain a copyright in said work.[41] Copyrightable subject matter consists of "original works of authorship fixed in any tangible medium of expression" which fall within eight enumerated categories.[42] Protectable subject matter is one of the primary distinguishing features between copyright and patent protection.[43] This is because copyright law protects creative works and expressions of art, whereas patent law protects inventions that are functional in nature.[44]

If a work qualifies as copyrightable subject matter under the Copyright Act and meets other judicially imposed criteria,[45] copyright automatically vests in the work's author.[46] The exclusive rights conferred by copyright are listed

35. *See* WILLIAM F. PATRY, COPYRIGHT LAW AND PRACTICE 26–30 (1994) (describing the enactment of the Copyright Act of 1790).

36. *Id.* at 88.

37. 17 U.S.C. § 102.

38. *Id.* § 106.

39. *Id.* § 302(a).

40. *See* Jessica D. Litman, *The Public Domain*, 39 EMORY L.J. 965, 975 (1990).

41. *See* Star Athletica, L.L.C. v. Varsity Brands, Inc., 137 S. Ct. 1002, 1008 (2017) ("A valid copyright extends only to copyrightable subject matter.").

42. 17 U.S.C. § 102(a) ("Works of authorship include . . . (1) literary works; (2) musical works, including any accompanying words; (3) dramatic works, including any accompanying music; (4) pantomimes and choreographic works; (5) pictorial, graphic, and sculptural works; (6) motion pictures and other audiovisual works; (7) sound recordings; and (8) architectural works.").

43. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 713–14 (2d Cir. 1992) (discussing whether software should be considered either copyrightable or patentable subject matter).

44. *See* Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH 305, 418 (2018). In addition to copyright, computer software can also attain protection within the United States through patent law. The scope of patent protection and its differences from copyright protection are beyond the focus of this Note. For more information regarding software patentability, see generally Bradford L. Smith & Susan O. Mann, *Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?*, 71 U. CHI. L. REV. 241 (2004). For information on modern software patentability requirements, see generally Ognjen Zivojnovic, *Patentable Subject Matter After Alice—Distinguishing Narrow Software Patents from Overly Broad Business Method Patents*, 30 BERKELEY TECH. L.J. 807 (2015).

45. *See, e.g.*, Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 346 (1991) (establishing a constitutional requirement of originality, which requires a copyrightable work to be an independent creation containing a modicum of creativity).

46. *See* 17 U.S.C. § 201(a).

in the Copyright Act and include the right to copy, create derivative works, distribute copies, and perform or display a work publicly.[47] Copyright holders may assign some or all of their exclusive rights to others.[48] One notable absence from the list of exclusive rights is the termination right, which instead constitutes a separate section of the Copyright Act.[49]

B. The Termination Right

The termination right allows original authors to regain their copyright over a work thirty-five years after they have sold, licensed, or otherwise transferred it.[50] This right is unique from other exclusive rights because it cannot be assigned or otherwise transferred to another,[51] and it can be exercised by the original author without the current owner's consent.[52] Although many contractual agreements contain language indicating a transfer of all rights held in a copyrighted work in perpetuity, such language is invalid insofar as it is applied to the termination right due to the inalienability provision.[53]

The motivation behind the termination right's creation was to better enable authors to reclaim works they sold during transactions of unequal bargaining power, particularly at the outset of their careers.[54] This sentiment is apparent from legislative history preceding the enactment of the Copyright Act, which recognized that "a provision safeguarding authors against unremunerative transfers . . . is needed because of the unequal bargaining position of authors, resulting in part from the impossibility of determining a work's value until it has been exploited."[55]

At the time, congressional concerns regarding unremunerative transfers were not merely hypothetical.[56] The creation of the famous comic book character Superman perhaps best illustrates the justifications for the termination right.[57] The creators of Superman sold the rights to their character to DC Comics in 1938 for a small sum of $130, yet the character went on to make billions of dollars for the company throughout its ongoing

47. *Id.* § 106.
48. *Id.* § 201(d).
49. *Id.* § 203(a)(3).
50. *Id.*
51. *Compare id.* § 203(a)(5) ("Termination of the grant may be effected notwithstanding any agreement to the contrary . . . ."), *with* § 201(d)(2) ("Any of the exclusive rights comprised in a copyright, including any subdivision of any of the rights specified by section 106, may be transferred.").
52. *See id.* § 203(a)(5); Richard D. Palmieri, Comment, *Who's the Author?: A Bright-Line Rule for Specially Commissioned Works Made for Hire*, 46 U. RICHMOND L. REV. 1175, 1176–77 (2012).
53. Palmieri, *supra* note 52, at 1176–77.
54. H.R. REP. NO. 94-1476, at 124 (1976).
55. *Id.*
56. *See, e.g.*, Dallas F. Kratzer III, Note, *Up, Up & Away: How Siegel & Shuster's Superman Was Contracted Away & DC Comics Won the Day*, 115 W. VA. L. REV. 1143, 1149–50 (2013).
57. *Id.*

existence.[58] The creators repeatedly tried to regain their copyright through litigation, yet they saw little success due to weak protections offered by the Copyright Act of 1909[59] ("1909 Act").

Under the 1909 Act, copyright duration only lasted for twenty-eight years, but was renewable for a second term, extending the overall duration to fifty-six years.[60] Both the renewal right and the termination right (as its successor) bifurcated the duration of copyright in order to allow original authors to reclaim ownership for the second duration.[61] However, there was one major difference between the two rights: the 1909 Act did not expressly make authors' renewal rights inalienable, and the U.S. Supreme Court subsequently ruled that "the Copyright Act of 1909 does not nullify agreements by authors to assign their renewal interests."[62] As a result, purchasers of copyrighted works under the 1909 Act had the ability to strong-arm authors into signing away the entire duration of their copyrights.[63] Inequitable scenarios of this sort are precisely what Congress sought to avoid through the addition of inalienable termination rights in the Copyright Act.[64]

Following the enactment of the Copyright Act, the Supreme Court recognized that the text "termination of the grant may be effected notwithstanding any agreement to the contrary" was clearly intended to rectify the impotency of an alienable termination right.[65] Thus, it is well established that the Copyright Act affords far greater protection against unremunerative transfers than did its previous iteration in the 1909 Act, largely due to the inalienability of the termination right.[66] However, while

58. *Id.*

59. Pub. L. No. 60-349, §§ 23–24, 35 Stat. 1075, 1080–81 (1909); *see* Kratzer, *supra* note 56, at 1151, 1160–62.

60. *See* Jorge L. Contreras & Andrew T. Hernacki, *Copyright Termination and Technical Standards*, 43 U. BALT. L. REV. 221, 232–33 (2014).

61. *Id.*

62. Fred Fisher Music Co. v. M. Witmark & Sons, 318 U.S. 643, 657 (1943).

63. *Cf.* Contreras & Hernacki, *supra* note 60, at 232–34 (explaining the congressional rationale behind making the termination right inalienable).

64. *See* H.R. REP. NO. 94-1476, at 124 (1976).

65. Mills Music, Inc. v. Snyder, 469 U.S. 153, 186 (1985) (White, J., dissenting) (quoting 17 U.S.C. § 203(a)(5)) ("[A]ssignees were able to demand the assignment of both terms at the time when the value of the copyrighted work was most uncertain. The termination provisions of the 1976 Act were designed to correct this situation. They guarantee to an author or his heirs the right to terminate a grant and any right under it 'notwithstanding any agreement to the contrary.'" (quoting 17 U.S.C. § 203(a)(5))).

66. *See id.* For a recent example of how the termination right has shifted the power dynamic between creators and copyright-reliant companies, see Brooks Barnes, *Disney Sues to Keep Complete Rights to Marvel Characters*, N.Y. TIMES (Sept. 27, 2021), https://www.nytimes.com/2021/09/24/business/media/disney-marvel-copyright-lawsuits.html [https://perma.cc/SY5B-LC4X] (explaining Disney's recent lawsuit to prevent copyright termination from being exercised by the heirs of some of its most famous comic book characters). Even companies as powerful as Disney have some cause for concern regarding copyright termination, especially given a recent victory for creators in the Second Circuit. *See* Horror Inc. v. Miller, No. 18-3123-CV, 2021 WL 4468980, at *19 (2d Cir. Sept. 30, 2021) (holding that the author of the original *Friday the 13th* screenplay was an

the rationale behind inalienable termination rights may have been sound as applied to most copyrightable works, it is likely to have unintended and arguably undesirable effects as applied to software.[67]

C. The Work Made for Hire Exception

An important caveat to the termination right is that it does not apply to works made for hire.[68] The term "work made for hire" is defined under § 101 of the Copyright Act as either one of two forms of work. The first form of work simply must be made by an employee "within the scope of his or her employment."[69] However, the second form of work contains three requirements: the work (1) is "specially ordered or commissioned," (2) falls within one of nine enumerated categories of works, and (3) is expressly agreed to be a work made for hire in a signed contract.[70]

Copyrights in works created under either definition of work made for hire do not vest in the author.[71] Instead, ownership and its associated rights vest in the author's employer.[72] In whom ownership initially vests is of utmost importance because ownership carries with it the exclusive rights granted under the Copyright Act, including the right to transfer and, by extension, the termination right.[73]

The two forms of works made for hire operate in different ways.[74] The clearest difference is that the first form specifically applies to employees, whereas the second can potentially apply to independent contractors, as well.[75] Although status as an employee is a necessary condition for the first form of work made for hire to apply, the Copyright Act does not define the word "employee."[76] This notable absence of a statutory definition led to the landmark case *Community for Creative Non-Violence v. Reid*,[77] in which the Supreme Court had to decide whether the creator of a commissioned statue was an independent contractor or an employee.[78] In conducting its analysis, the Court began with the statutory definition of work made for hire under § 101.[79] Given that statues are not listed among the nine categories within

independent contractor at the time of its creation and could exercise his termination right as a result).

67. *See infra* Part II.

68. 17 U.S.C. § 201(b).

69. *Id.* § 101(1).

70. *Id.* § 101(2) (listing the nine categories as "a work specially ordered or commissioned for use as a contribution to a collective work, as a part of a motion picture or other audiovisual work, as a translation, as a supplementary work, as a compilation, as an instructional text, as a test, as answer material for a test, or as an atlas").

71. *See* Matthew R. Harris, Note, *Copyright, Computer Software, and Work Made for Hire*, 89 MICH. L. REV. 661, 662 (1990).

72. *Id.*

73. 17 U.S.C. § 201(a), (d).

74. *See* Harris, *supra* note 71, at 667–68.

75. *See id.*

76. *Id.* at 663.

77. 490 U.S. 730 (1989).

78. *Id.* at 753.

79. *Id.* at 732.

the second definition[80] and that there was no signed agreement between the parties, the Court instead turned to the first definition of work made for hire.[81]

The Court adopted a common law of agency meaning of employment and, in doing so, established twelve nonexclusive factors that indicated whether a given work is the product of an employee or independent contractor relationship.[82] These factors were drawn from the *Restatement (Second) of Agency*[83] and included, among other factors, the level of skill required, location of performance of the work, owner of the tools in use, duration of the work, and degree of control each party has over determining when and how long to work.[84] The Court noted the fact-intensive nature of such an inquiry and specified that none of the factors are determinative in isolation.[85] Thus, understanding how the *Reid* factors are likely to apply to programmers' creation of software is best accomplished by reviewing an appropriately analogous case.

D. Software as Copyrightable Subject Matter

The history of software development in the United States has been quite brief, especially considering the substantial length of copyright duration.[86] Although the first programming languages were published around 1957, it was not until the mid-1980s that customer-facing software became popular among the public through the personal computer and computer programs such as Microsoft Word.[87] Further innovation came about in the 1990s with the inception of OSS and the World Wide Web, and continued into the 2000s with the rise of social media and the invention of smartphones.[88]

80. *See* 17 U.S.C. § 101(2).

81. *Reid*, 490 U.S. at 736 ("[S]culpture is not one of the nine categories of works enumerated in that subsection, and the parties had not agreed in writing that the sculpture would be a work for hire.").

82. *See id.* at 751–52. The Court also established a second requirement: if the hired party was found to be an employee, the work must have been made within the scope of employment. *See id.* at 739–41. This requirement is not relevant to this Note's analysis, but to learn about its application, see generally James B. Wadley & JoLynn M. Brown, *Working Between the Lines of* Reid*: Teachers, Copyrights, Work-For-Hire and a New Washburn University Policy*, 38 WASHBURN L.J. 385 (1999).

83. RESTATEMENT (SECOND) OF AGENCY § 220(2) (AM. L. INST. 1999).

84. *Reid*, 490 U.S. at 751–52 (listing the remaining factors as "whether the hiring party has the right to assign additional projects to the hired party; . . . the method of payment; the hired party's role in hiring and paying assistants; whether the work is part of the regular business of the hiring party; whether the hiring party is in business; the provision of employee benefits; and the tax treatment of the hired party." (footnotes omitted)).

85. *Id.* at 752.

86. *See* 17 U.S.C. § 302(a) (establishing the general copyright duration as the remainder of the author's life plus seventy years).

87. *See* Micah Yost, *A Brief History of Software Development*, MEDIUM (Jan. 25, 2018), https://medium.com/@micahyost/a-brief-history-of-software-development-f67a6e6ddae0 [https://perma.cc/2LE2-27D4].

88. *See id.*

Legislative history indicates that, in passing the Copyright Act, Congress was considering including software as copyrightable subject matter.[89] However, before doing so, Congress created the National Commission on New Technological Uses of Copyrighted Works (CONTU) to find issues pertaining to the inclusion of software within copyright law.[90] CONTU issued a final report to Congress containing its recommendations, including an amendment to the Copyright Act titled: "§ 117: Limitations on Exclusive Rights: Computer Programs."[91] Although the CONTU report did not mention termination rights, the limitations contained in § 117 implicate termination indirectly.[92]

Even following Congress's enactment of CONTU's recommendations, the extent to which computer programs were considered protectable subject matter under the Copyright Act remained unclear.[93] However, the Third Circuit's holding in *Apple Computer, Inc. v. Franklin Computer Corp.*[94] affirmed the now-predominant legal understanding of software as copyrightable subject matter.[95] The court held that software is protectable subject matter because the code by which it operates is a "literary work" under § 101 of the Copyright Act.[96] The court based its ruling on legislative history, including CONTU's creation and Congress's subsequent adoption of CONTU's findings.[97]

Although the legislative history was supportive of affording copyright protection to computer programs as a whole, it provided little indication as to the copyrightability of object code as distinguished from source code.[98] Source code constitutes instructions written by programmers in various programming languages that are readable by humans.[99] Object code, on the other hand, is the output of the source code that takes form as a series of ones and zeros that, while appearing incomprehensible to humans, can be read by computers and translated into an executable action.[100] Overall, the court's holding was important in clarifying that both source and object code are

89. *See* H.R. REP. NO. 94-1476, at 53–54 (1976) ("It also includes computer data bases, and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves.").

90. Act of Dec. 31, 1974, Pub. L. No. 93-573, tit. II, 88 Stat. 1873, 1873–75.

91. *Computers and Copyright: Recommendations for Statutory Change*, *in* FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGY USES OF COPYRIGHTED WORKS (1978) [hereinafter CONTU Report], http://digital-law-online.info/CONTU/contu6.html [https://perma.cc/9DA2-4R49].

92. *See infra* Part II.D.

93. *See infra* note 98 and accompanying text.

94. 714 F.2d 1240 (3d Cir. 1983).

95. *Id.* at 1253–54.

96. *Id.* at 1249.

97. *Id.* at 1248–49.

98. *See id.* at 1246 (explaining that the district court had found congressional intent regarding the copyrightability of object code to be unclear).

99. *See id.* at 1243.

100. *See id.*

considered "literary works," thus broadening the understanding of the definition contained in the Copyright Act.[101]

E. Practical Implications on the Software Industry

The practical implications of software terminability may not be immediately apparent given that computer programs tend to retain little market value after thirty-five years of technological development.[102] This has led one commentator to doubt the practical relevance of software terminability altogether.[103] Yet, this view fails to appreciate that the overall function of a program is tied to its constituent building blocks of code,[104] which may still be in use in various ways even if written decades earlier.[105]

Although the market value of software is ultimately tied to the overall function of the program,[106] functionality is in turn dependent on the operability of the program's code. For programs to remain operational, they often require maintenance in the form of bug fixes.[107] Furthermore, because nonessential alterations to code constitute making a derivative work, maintenance and bug fixes may be copyright infringement if performed in the absence of the copyright owner's permission.[108] Thus, by shifting focus away from the value of the overall program and toward the code from which it is made, the true potential value of termination becomes more apparent.[109]

Source code can take a variety of forms through the "language" in which it is written and can perform different functions as a result.[110] One such function is the operation of internal computer systems, as opposed to outward customer-facing software like Microsoft Word.[111] Internal computer software tends to be replaced infrequently due to the associated costs of doing

101. *See id.* at 1249 ("[T]he category of 'literary works', one of the seven copyrightable categories, is not confined to literature in the nature of Hemingway's *For Whom the Bell Tolls*. The definition of 'literary works' in section 101 includes expression not only in words but also 'numbers, or other . . . numerical symbols or indicia', thereby expanding the common usage of 'literary works.'").

102. *See* Harris, *supra* note 71.

103. *See id.*

104. The Second Circuit has developed a three-part test to apply to computer software in an effort to separate protectable expression from the uncopyrightable elements of a program, such as ideas, code that is already in the public domain, and functional code. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706–712 (2d Cir. 1992). However, this test is only applicable once an infringement suit has been brought; thus, it will be inapplicable in a termination context without another program with which to compare it. For more information on the Second Circuit's test, see *id.*

105. *See infra* notes 113–17 and accompanying text.

106. *See* Yang, *supra* note 18, at 188.

107. *See* Phelps, *supra* note 19, at 270.

108. *Id.*

109. *Id.*

110. *See* Whelan Assocs. v. Jaslow Dental Lab'y, Inc., 797 F.2d 1222, 1230–31 (3d Cir. 1986).

111. *See generally* Credit for Increasing Research Activities, 81 Fed. Reg. 68,299 (Oct. 4, 2016) (to be codified at 26 C.F.R. pt. 1) (defining internal use software).

so, yet it still requires bug fixes and periodic updates.[112] This dynamic makes internal software a potential hotspot for termination issues, as updates to the software could constitute infringement if attempted on a terminated program.[113]

1. Internal Computer Software

One need not look far to recognize the potential implications of software termination. The programming language known as COBOL, which stands for common business-oriented language, powered 43 percent of banking systems as of 2017[114] and continues to be widely used by U.S. federal and state governments.[115] Despite its prevalence, COBOL is roughly sixty-years-old and has been considered by programmers since the 1980s to be an obsolete "dead language."[116] Although plenty of new languages have emerged that are more complex and efficient than COBOL, efforts to overhaul internal software operations in order to move away from COBOL have proven to be extremely difficult and expensive.[117] The persistence of COBOL is indicative of the reality that internal software operations are not in a state of constant innovation like much of the rest of the software industry.[118]

Despite the stagnation of internal software systems, they still require periodic updates, bug fixes, and occasional significant alterations.[119] Updates often require modifying the source code, which itself requires the exclusive right to create derivative works held under copyright.[120] Thus,

112. *See* John Blyler, *COBOL Coders Needed for Coronavirus Fight*, DESIGN NEWS (Apr. 23, 2020), https://www.designnews.com/design-hardware-software/cobol-coders-needed-coronavirus-fight [https://perma.cc/ZR6E-E5VW] (explaining how major updates to computer systems were needed both in the lead-up to Y2K and in the wake of the coronavirus pandemic).

113. *See* Phelps, *supra* note 19, at 270.

114. *COBOL Blues*, REUTERS GRAPHICS, http://fingfx.thomsonreuters.com/gfx/rngs/USA-BANKS-COBOL/010040KH18J/index.html [https://perma.cc/EJ2U-HXLM] (last visited Oct. 29, 2021).

115. Mark Sullivan, *COBOL, A 60-Year-Old Computer Language, Is in the COVID-19 Spotlight*, FAST CO. (Apr. 10, 2020), https://www.fastcompany.com/90488862/what-is-cobol [https://perma.cc/A6SA-TLFX].

116. Charles R. Martin, *Brush Up Your COBOL: Why Is a 60 Year Old Language Suddenly in Demand?*, OVERFLOW (Apr. 20, 2020), https://stackoverflow.blog/2020/04/20/brush-up-your-cobol-why-is-a-60-year-old-language-suddenly-in-demand/ [https://perma.cc/F8FU-TJ3U] ("By the 80's, students were being told that COBOL was a dead language, and no one was studying it any more. Now, in 2020, governments and banks are pleading for COBOL programmers, the language that wouldn't die.").

117. Anna Irrera, *Banks Scramble to Fix Old Systems as IT 'Cowboys' Ride into Sunset*, REUTERS (Apr. 11, 2017, 9:42 AM), https://www.reuters.com/article/us-usa-banks-cobol/banks-scramble-to-fix-old-systems-as-it-cowboys-ride-into-sunset-idUSKBN17C0D8 [https://perma.cc/9ZU7-RW9Z] ("Commonwealth Bank of Australia, for instance, replaced its core banking platform in 2012 with the help of Accenture and software company SAP SE. The job ultimately took five years and cost more than 1 billion Australian dollars ($749.9 million).").

118. *See supra* notes 87–88 and accompanying text.

119. *See supra* note 112 and accompanying text.

120. *See* Phelps, *supra* note 19, at 270.

altering the source code in the absence of copyright could risk copyright infringement.[121] COBOL is a prime example of the potential longevity of internal software, and there is no telling which of the modern programming languages used today may still be relied on decades from now. Therefore, the application of termination rights to software has the potential to place obstacles of significant monetary cost in front of internal software development.

2. Open Source Software

Similar concerns arise in the OSS context. The open source movement was founded and promoted based on a unique set of principles that were intended to maximize accessibility of source code to the public at large.[122] The policy goals underlying the movement were summarized by two scholars as promoting security, affordability, transparency, perpetuity, interoperability, flexibility, and localization.[123] These goals are maintained by licensing agreements that every user must agree to prior to using the provided source code.[124]

Although open source licenses come in various forms,[125] they all permit modification and redistribution of the original program, as long as the user abides by explicit conditions.[126] These conditions are often referred to by the term "copyleft," in order to contrast the open nature of OSS with the proprietary interests associated with copyright.[127] A representative example of such conditions is the General Public License (GPL), a widely used form of license for OSS, which requires perpetual licensing, under the same terms, of any subsequent distribution of either the software itself or a derivative work created using its source code.[128] Continued open access to the source code of both the original software and any additional code present in the derivative work is a fundamental requirement for a GPL.[129] The Federal Circuit has recognized the legitimacy of these licenses, holding that use of OSS that does not comply with the licensing agreement constitutes copyright infringement.[130]

The open source movement has found great success as hundreds of millions of people use open source programs such as Linux, Firefox, and

121. *Id.*
122. *See generally* RICHARD M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN (Joshua Gay ed., 1st ed. 2002).
123. Tony Casson & Patrick S. Ryan, *Open Standards, Open Source Adoption in the Public Sector, and Their Relationship to Microsoft's Market Dominance*, *in* THE STANDARDS EDGE: UNIFIER OR DIVIDER? 87, 91 (Sherrie Bolin ed., 2006).
124. *See* Armstrong, *supra* note 13, at 383–84.
125. *See* Arnoud Engelfriet, *The Best of Both Worlds*, INTELL. ASSET MGMT. MAG., Aug.–Sept. 2006, at 37, 37–38.
126. *See* Armstrong, *supra* note 13, at 383–84.
127. *Id.* at 372.
128. *See* Phelps, *supra* note 19, at 263.
129. *Id.*
130. Jacobsen v. Katzer, 535 F.3d 1373, 1381 (Fed. Cir. 2008).

Wikipedia.[131] Yet, with this success comes a significant amount of reliance on the continuation of OSS programs.[132] Potential exercise of termination rights could threaten such continuation, especially because termination is effective not only on the original grant but on all subsequent licenses, as well.[133] Therefore, even derivative programs that may hardly resemble the original are still hindered by termination as long as they retain some of the original source code.[134] This is problematic given that creation of derivative works is not only possible under open source licenses but is encouraged as one of the core beliefs underlying the open source model.[135] Furthermore, practical difficulties arise when attempting to separate the original author's contributions from the modern version containing decades of additions and alterations.[136]

Perhaps even more damaging to the open source movement would be the original authors' ability to legally violate their own guarantee of perpetuity.[137] GPLs expressly state that "[a]ll rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met."[138] Yet, despite its expressed irrevocability, termination rights are inalienable even by signed contract.[139] As a result, original authors of OSS programs reserve the right to terminate all licenses after thirty-five years, despite their expression to the contrary.[140] One commentator has expressed concern that exercise of termination in such a context would "present a clear affront to the community norms of nonproprietization and mutual sharing that characterize a number of the most vibrant open-content projects."[141]

The likelihood of an original author exercising this right may appear slim due to an initial willingness to contribute one's program to the public in lieu of monetary gain. However, upon the original author's death, termination rights are transferred to the author's surviving spouse, children, grandchildren, executor, administrator, personal representative, or trustee.[142] Naturally, there is no guarantee that a beneficiary will share the same altruistic motives as the original author.[143] Concerns of a potentially rogue beneficiary are slightly tempered by the temporal limitation contained within

131. *See* Armstrong, *supra* note 13, at 361.
132. *See* Phelps, *supra* note 19, at 262.
133. *See* Mills Music, Inc. v. Snyder, 469 U.S. 153, 165 (1985).
134. *See* Armstrong, *supra* note 13, at 407.
135. *See* Stallman, *supra* note 122, at 203 ("Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.").
136. *See* Armstrong, *supra* note 13, at 363.
137. *Id.* at 405.
138. *GNU General Public License*, FREE SOFTWARE FOUND. (June 29, 2007), http://www.gnu.org/licenses/gpl.html [https://perma.cc/2L5F-CVQP].
139. *See* 17 U.S.C. 203(a)(5).
140. *See* Armstrong, *supra* note 13, at 374.
141. *Id.* at 363.
142. 17 U.S.C. § 203(a)(2).
143. *See* Armstrong, *supra* note 13, at 362 (describing a hypothetical worst-case scenario along these lines).

the termination statute, as the termination right can only be exercised within a window of five years.[144] Yet, considering that the program is presumably now known to appeal to a sizable market and that the original author's goodwill may not necessarily transfer along with their termination right, OSS users cannot be truly secure in their perpetual access to the program until the termination window has elapsed.

3. Proprietary Software

Lastly, proprietary software is also vulnerable to threats posed by termination.[145] Given that the source code within companies' customer-facing products is kept secret from the public, it is difficult to estimate the amount of code that is transferred between new iterations of the product.[146] However, as evident from OSS, programs from as many as twenty years ago still retain a significant amount of original source code despite thousands of programmers constantly altering and improving the programs.[147] Thus, even well-established programs are at risk of being stalled by the potential exercise of termination rights depending on the amount of source code retained from one iteration to the next.

Additionally, proprietary software designed for entertainment purposes may actually retain value inherent in the program itself. One prominent example is the video game industry, which is replete with examples of old games that have been discontinued in production yet later face a resurgence of demand.[148] Thus, video games could prove to be a valuable asset for the games' original programmers to terminate and monetize for their own benefit.[149]

Large-scale works also fall within the category of proprietary software. Sometimes referred to as "big code," large-scale works include programs such as Windows 10, which contains an estimated fifty million lines of code.[150] Big code is not limited to the software industry, as it is also utilized

144. 17 U.S.C. § 203(a)(3).

145. *See* Phelps, *supra* note 19, at 272 ("This interpretation, however, would lead to problematic results even when applied to traditional commercial software.").

146. *See* Sonia K. Katyal, *The Paradox of Source Code Secrecy*, 104 CORNELL L. REV. 1183, 1207 (2019).

147. *See* Phelps, *supra* note 19, at 270 ("[C]onsidering that fourteen years after its inception, Linus Torvalds was still 'one of the main contributors to the Linux kernel project' it is not inconceivable that a substantial portion of his original code will remain after another twenty-one years; therefore, the original license may be subject to termination." (quoting Ilkka Tuomi, *Evolution of the Linux Credits File: Methodological Challenges and Reference Data for Open Source Research*, 9 FIRST MONDAY, June 2004)).

148. *See generally* Sean F. Kane, *Copyright Assignment Termination After 35 Years: The Video Game Industry Comes of Age*, PILLSBURY (Sept. 23, 2013), https://www.jdsupra.com/legalnews/copyright-assignment-termination-after-3-37405/ [https://perma.cc/FF63-4KVV].

149. *See id.*

150. Christopher Tozzi, *Code Challenges: Coping in the Era of 'Big Code,'* ITPROTODAY (Aug. 10, 2020), https://www.itprotoday.com/devops-and-software-development/code-challenges-coping-era-big-code [https://perma.cc/48QR-ZNXY].

in industries ranging from telecommunications to transportation[151] (exemplified by the fact that a standard car from 2012 relies on approximately one hundred million lines of code).[152] Naturally, creating works of this scale requires a collaborative effort by a substantial number of programmers.[153] If each programmer's independent contribution of code were to be copyrightable, the effects of termination on big code could be debilitating to the software industry and the economy as a whole. Predicting how courts are likely to treat the exercise of termination rights in large-scale software works will help inform an effective solution to the foregoing issues in proprietary software.

II. PREDICTING THE SCOPE AND SEVERITY OF THE EFFECTS OF SOFTWARE TERMINABILITY

Software terminability has yet to be addressed by the courts, thus leaving the current state of the law uncertain.[154] Clarifying the law first requires determining the scope of the problem, namely, how many software works will be vulnerable to termination. Determining the scope requires analysis of the work made for hire exception under the Copyright Act, as well as limitations under relevant case law relating to works created by multiple authors. In addition to scope, it is also necessary to determine the extent to which the exercise of termination rights will actually affect vulnerable software.

Part II addresses both sides of the impending conflicts stemming from software terminability. Part II.A addresses the question of whether courts will find the termination right applicable to software by extension of its inclusion as a literary work under the Copyright Act. Part II.B considers the extent to which programmers' termination rights may be stifled by the work made for hire exception. Part II.C examines whether large-scale software works will be at risk due to the exercise of termination rights. Finally, Part II.D analyzes § 117 of the Copyright Act to determine the extent to which it may mitigate problems posed by software termination.

A. Will the Courts Find Software to Be Terminable?

The threshold question that must be addressed is whether courts will find the termination right applicable to software. In one respect, this is an open question because no court has ruled on the issue.[155] However, there is no discernible language within the Copyright Act that would specifically

151. *See* SOURCEGRAPH, THE EMERGENCE OF BIG CODE 14 (2020), https://info.sourcegraph.com/hubfs/CTA%20assets/sourcegraph-big-code-survey-report.pdf [https://perma.cc/7D9J-SNVV].
152. *See* Tozzi, *supra* note 150.
153. *See* Harris, *supra* note 71, at 694–95.
154. *See* Armstrong, *supra* note 13, at 422.
155. *See id.*

exempt software from the termination right.[156] Furthermore, following software's inclusion as a literary work, courts have created common law rules when rules are necessary to treat software consistently with all other forms of copyrightable work.[157]

The primary distinguishing feature of software within the Copyright Act is that computer programs have limited exceptions to exclusive rights under § 117.[158] Despite having a separate section of exemptions, the termination right is not included among them.[159] Termination's absence is particularly notable given that Congress specifically created CONTU to find issues pertaining to copyright in software.[160] While CONTU made numerous recommendations to Congress, including the adoption of § 117, termination was not specifically addressed.[161]

The lack of a statutory exception for software likely renders terminability inevitable.[162] However, opposing litigators may have strong policy arguments at their disposal.[163] First, software's absence from the Copyright Act is arguably indicative of Congress's lack of intent for termination to extend to software.[164] Both the termination right and the inclusion of software as a copyrightable work were new additions to the law and did not immediately interact with each other.[165] Therefore, it is unlikely that Congress anticipated the consequences of termination on software thirty-five years in the future.[166] Strengthening this claim is the apparent incongruence between the policy justifications underlying the termination right and its likely effects when applied to software.[167]

The constitutional justification for copyright protection is to further artistic progress; yet software termination arguably impedes this goal. Software differs from other copyrightable works because its creative element of written code is tied to a functional product in the overall software.[168] Termination of a computer program essential to a particular business could paralyze operations, causing functional harm that far exceeds protection of

156. Pub. L. No. 94-553, 90 Stat. 2541 (codified as amended in scattered sections of the U.S.C.).

157. *See, e.g.*, Comput. Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 706–07 (2d Cir. 1992) (establishing a test to separate idea from expression within a computer program so as to treat a computer program consistently with other copyrightable works).

158. *See generally* 17 U.S.C. § 117.

159. *See id.*

160. Act of Dec. 31, 1974, Pub. L. No. 93-573, 88 Stat. 201.

161. *See* CONTU Report, *supra* note 91.

162. Commentators appear to implicitly presume that termination will apply to software. *See generally*, Armstrong, *supra* note 13; Phelps, *supra* note 19.

163. *See* Armstrong, *supra* note 13, at 362.

164. *See id.* at 416–17.

165. *See id.* at 422.

166. *See id.* at 416–17.

167. *See id.*

168. *See* Google LLC v. Oracle Am., Inc., 141 S. Ct. 1183, 1198 (2021) ("Generically speaking, computer programs differ from books, films, and many other 'literary works' in that such programs almost always serve functional purposes."); LOPEZ, *supra* note 20, at 7 ("The controversy is linked to the unique nature of computer software that performs technical functions through creative expression.").

the software's creative elements, which copyright was intended to secure. This dynamic is even more pronounced in OSS, as the open accessibility of code has been the source of substantial progress within the field of programming.[169] Termination of OSS works would have detrimental effects both in the immediate and long term, as it would "shrink the commons" of publicly available code while also deterring future reliance by OSS users who could no longer trust the perpetuity of even the most explicit licensing agreements.[170]

Furthermore, the policy concerns regarding unremunerative transfers that necessitated the termination right are arguably not as salient as applied to proprietary software programmers and are wholly absent from programmers of OSS.[171] Programmers create software to perform a specific function, and they do not begin writing code without some idea of what this function will be.[172] From the program's function, those in the field of work can reasonably discern its value based on the relevant markets' needs.[173] The ability to assess the market value of a computer program enables programmers to negotiate for reasonably fair value at the time of transaction.[174] On the other hand, authors of other copyrightable works, such as art and music, often create work without knowing its ultimate value.[175] When market value is speculative, the amount purchasers are willing to pay is unlikely to reflect the fair value of works that become wildly successful.[176]

The underlying dispute between the "objective" value of software through its function and the "subjective" value of most other copyrightable works through their creativity has existed for decades and has implications well beyond issues related to termination.[177] However, differences between the two types of works are relevant to software terminability, as the "hapless creator" Congress intended to protect likely does not exist to the same degree in the software market.[178]

Constitutional and congressional intent arguments against software terminability are not without merit. However, courts in a textualist era are unlikely to read an exception for software in the absence of any statutory

169. *See* Jacobsen v. Katzer, 535 F.3d 1373, 1378–79 (Fed. Cir. 2008) (describing the benefits of open access, which have enabled new forms of software development).

170. *See* Armstrong, *supra* note 13, at 363, 374.

171. *See* Harris, *supra* note 71, at 699.

172. *Id.* ("The paradigm of the artist creating art for art's sake—and later learning of the need to protect the work—does not appear relevant here.").

173. *Id.* at 688, 697, 699.

174. *Id.* at 688, 698.

175. *See* Ian McClure, *Termination Rights: A Second Bite at the Apple*, FED. LAWYER, Jan. 2009, at 16, 16, http://www.ipprospective.com/wp-content/uploads/2009/01/termination-rights2.pdf [https://perma.cc/PM79-AVHF] ("Most intellectual property is difficult to value before products embodying the rights are sold on the market. Accurately pricing an exclusive license to use intellectual property is arbitrary at best.").

176. *See id.*

177. *See* Harris, *supra* note 71, at 697 ("Computer software itself differs significantly from other copyrightable subject matter. For example, much of the value of software comes from its utilitarian rather than its aesthetic aspects.").

178. *Id.* at 699.

basis for doing so.[179] While policy arguments likely will not defeat software terminability at the threshold question of applicability, they could prove to be effective in arguing for other judicially imposed limitations or future legislative amendments.

B. Will the Courts Find the Works of Freelance Software Programmers to Be Works Made for Hire?

The largest obstacle for programmers who seek to reclaim ownership of their works is the explicit exclusion of works made for hire.[180] The Copyright Act defines two types of works made for hire,[181] yet it is unclear how these definitions will apply to software.

The first definition is broader in scope because it looks to the nature of the relationship between contracting parties rather than the type of work being created.[182] Thus, under this definition, software is just as susceptible to being a work made for hire as any other copyrightable work.[183] On the other hand, the second definition is limited to a select number of specified categories, and computer software is not among them.[184] Yet, if courts were to find that computer software does fit within this definition, the effects would be far more consequential, as employers could effectively contract around the termination right.[185]

1. Statutory Definition #1

The first definition under § 101 hinges on the meaning of the word "employee," which was not defined by Congress within the Copyright Act but was later clarified by the Supreme Court in *Community for Creative Non-Violence v. Reid*.[186] The Court established the relevant factors used to determine whether a hired party was an employee or independent contractor.[187] Predicting how the *Reid* factors would apply to programmers' creation of computer software is a difficult task, given the fact-intensive nature of the inquiry. However, in the wake of *Reid*, an informal hierarchy of the *Reid* factors has emerged as a result of the factors' application by the courts. The Second Circuit went as far as expressly stating the five factors it found to be most frequently relevant and that should be weighed more heavily as a result.[188] Relying on existing case law, one scholar has conducted a comprehensive study ("Vacca study") in order to rank the *Reid*

179. *See* Armstrong, *supra* note 13, at 422–23.
180. 17 U.S.C. § 203(a)(3).
181. *Id.* § 101 (defining "work made for hire").
182. *See* Harris, *supra* note 71, at 677–78.
183. *See* 17 U.S.C. § 101(1) (requiring only that the work is "prepared by an employee within the scope of his or her employment").
184. *See id.* § 101(2).
185. *See id.* § 203(a).
186. 490 U.S. 730, 738–40 (1989).
187. *Id.* at 750–52.
188. *See* Aymes v. Bonelli, 980 F.2d 857, 861 (2d Cir. 1992) (listing five of the *Reid* factors that will be relevant in nearly every case).

factors by overall importance.[189] These aggregate analyses are likely the best available method of determining the impact of the *Reid* factors on the software industry as a whole.

The Vacca study splits the fourteen *Reid* factors into five subcategories of increasing importance.[190] The most important category of factors includes the taxes paid by the hiring party, whether the hired party receives employee benefits, and whether the employee is paid by time or by completion of the work.[191] Unfortunately, these factors are too fact-specific to be applied to the work done by programmers in the aggregate without a specific fact pattern to analyze. However, results indicate that in cases involving a formal, salaried employment relationship, courts are very likely to find the author to be an employee.[192] Therefore, it is intuitive that the programmers of most interest in a termination analysis are those who do not receive a formal salary.

The second most important group includes two factors that may be aggregated: (1) the skill required for the work and (2) the source of instrumentalities and tools from which the work is created.[193] Programmers' work typically requires a bachelor's degree in computer science or software engineering.[194] Although a significant amount of programmers identify as fully or partially self-taught,[195] courts have consistently found computer programming to be a skilled occupation.[196] Overall, the expertise required weighs in favor of programmers being more likely to be deemed independent contractors than employees. The remaining factors—location of the work performed and tools used in completing the work—depend on similar facts. Intuitively, programmers who work at home are far more self-reliant in terms of the tools used than are programmers who work in an office setting. Although these factors are less determinative,[197] they likely weigh in favor of an independent contractor relationship given that programmers are among the occupations most likely to work from home.[198]

189. *See generally* Ryan Vacca, *Works Made for Hire—Analyzing the Multifactor Balancing Test*, 42 FLA. ST. U. L. REV. 197 (2014).

190. *Id.* at 229.

191. *Id.*

192. *See id.* at 234.

193. *Id.* at 229.

194. *Occupational Outlook Handbook: Computer Programmers*, U.S. BUREAU OF LABOR STATISTICS, https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm#tab-4 [https://perma.cc/C5FY-C9XS] (last visited Oct. 29, 2021).

195. Karen Turner, *Lots of Coders Are Self-Taught, According to Developer Survey*, WASH. POST (Mar. 30, 2016), https://www.washingtonpost.com/news/the-switch/wp/2016/03/30/lots-of-coders-are-self-taught-according-to-developer-survey/ [https://perma.cc/V6UA-6ANG].

196. *See, e.g.*, Aymes v. Bonelli, 980 F.2d 857, 862 (2d Cir. 1992); MacLean Assocs. v. Wm. M. Mercer-Meidinger-Hansen, Inc., 952 F.2d 769, 777 (3d Cir. 1991).

197. *See* Vacca, *supra* note 189, at 229. These factors may carry even less weight today considering that the COVID-19 pandemic has greatly increased the number of U.S. workers who work from home. *See* May Wong, *Stanford Research Provides a Snapshot of a New Working-from-Home Economy*, STANFORD NEWS (June 29, 2020), https://news.stanford.edu/2020/06/29/snapshot-new-working-home-economy/ [https://perma.cc/E4NS-4Q7R].

198. Christopher Groskopf, *For Programmers, the Ultimate Office Perk Is Avoiding the Office Entirely*, QUARTZ (Apr. 12, 2017), https://qz.com/950973/remote-work-for-

Application of the few aggregable *Reid* factors to programmers' work appears to indicate that programmers are slightly more likely to be deemed independent contractors than employees.[199] However, most of the *Reid* factors are case-specific and thus difficult to predict, including the three factors that, according to the Vacca study, make up the most important category.[200] An example of how a court would apply the *Reid* factors to a specific instance of a programmer's work is the case of *JustMed, Inc. v. Byce*.[201] Although the Ninth Circuit ultimately upheld the district court's finding that the creator of the software at issue was an employee,[202] the court's reasoning is the most noteworthy portion of the case. When considering each of the *Reid* factors, the court placed less weight on those that were attributable to the "nature of the business and the work."[203] This included factors such as the programmer's level of skill, ability to work from home, and ability to set his own hours, all of which are typically indicative of an independent contractor relationship.[204] The court analyzed each factor "in light of the kind of work" the programmer was doing, which in effect nullified the generally applicable factors that tend to weigh in favor of programmers.[205] By extension, the court afforded the more particular, case-specific factors more importance than they would otherwise have.[206]

The court's holding in *JustMed* is by no means preclusive to programmers' termination rights, as the case-specific factors may still weigh in their favor. However, the holding makes the *Reid* factors' outcome less predictable as applied to software works. This is because the court's contextual analysis effectively nullifies the advantageous factors programmers would otherwise have by virtue of their line of work.[207] The unpredictability of the multifactor *Reid* analysis is undesirable for both the hiring party and the worker, as neither can feel entirely secure in their ownership rights.[208] Furthermore, there are additional practical issues in cases arising from the exercise of termination given that the facts at issue are more difficult to establish after thirty-five years have passed.[209] Overall, the first definition of work made for hire fails to provide a clear estimate as to the scope of works that will fall within the work made for hire exception from termination.

programmers-the-ultimate-office-perk-is-avoiding-the-office-entirely/ [https://perma.cc/5WMM-NDHP].

199. *See supra* notes 193–98 and accompanying text.
200. *See* Vacca, *supra* note 189, at 229.
201. 600 F.3d 1118 (9th Cir. 2010).
202. *Id.* at 1128.
203. *Id.* at 1127.
204. *See id.*; *see also supra* notes 197–98 and accompanying text.
205. *See JustMed*, 600 F.3d at 1128.
206. *Cf. id.*
207. *See supra* notes 197–201 and accompanying text.
208. *See* Harris, *supra* note 71, at 682.
209. *See* Scorpio Music S.A. v. Willis, No. 11CV1557, 2013 WL 790940, at *5 (S.D. Cal. Mar. 4, 2013) ("Policy arguments can also be made for avoiding the filing of lawsuits decades after the creation of a work, when witnesses may be dead, documents lost, and memories faded.").

2. Statutory Definition #2

The second definition under § 101 defines works made for hire on the basis of the type of work at issue, as opposed to the nature of the employment relationship under the first definition. In doing so, the second definition lists nine categories under which a work must fall to qualify as a work made for hire.[210] Computer software is not included in the listed categories.[211] Given courts' general reluctance to stray beyond the specified categories, this absence has been interpreted by some scholars as an indication that computer software can only be classified as a work made for hire under the first definition.[212] However, courts have recently been more willing to interpret the categories expansively enough to include computer software, either in whole or in part, depending on which aspect of the program was at issue. For example, courts have considered software programs as a whole to be "compilations,"[213] segments of source code to be a "contribution to a collective work,"[214] and the nonliteral elements to qualify as an "audiovisual work."[215]

If an expansive interpretation of the second definition becomes widely accepted, it could drastically reduce programmers' future ability to exercise their termination right. This is due to the third prong of the definition, which requires an express agreement signed by the parties that the work is to be considered a work for hire.[216] In effect, the hiring party could circumvent the inalienability of termination rights by ensuring that the rights never vest in the programmer to begin with.[217] To do so, an individual or company commissioning the work would have to include a provision specifying that the work is to constitute a work for hire, which the programmer must then sign.[218] Thus, to avoid running afoul of the inalienability provision of § 203 by assigning the software itself through contract, the hiring party could instead bind the programmer in a work made for hire relationship, such that any work created from that point onward would vest in the hiring party.

As the law currently stands, the second definition of work made for hire presents a potential avenue for employers to circumvent independent contractors' termination rights. However, there are two primary difficulties faced by proponents of an expansive interpretation of § 101. First, there is an apparent lack of textual basis for including software in the nine categories,

210. 17 U.S.C. § 101(2).
211. *Id.*
212. *See* Harris, *supra* note 71, at 685; Phelps, *supra* note 19, at 265.
213. Stanacard, LLC v. Rubard, LLC, No. 1:12-CV-5176, 2016 U.S. Dist. LEXIS 15721, at *22 (S.D.N.Y. Feb. 3, 2016).
214. iXL Inc. v. Adoutlet, No. 01 C 0763, 2001 U.S. Dist. LEXIS 3784, at *27 (N.D. Ill. Mar. 29, 2001).
215. Breadmore v. Jacobson, No. 4:13-CV-361, 2014 U.S. Dist. LEXIS 97332, at *17 (S.D. Tex. July 14, 2014).
216. 17 U.S.C. § 101(2).
217. *See* 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 11.02[A][2] (1998).
218. *See supra* note 70 and accompanying text.

which may immediately end any effort to do so before a textualist court.[219] Second, the legislative history of the Copyright Act makes no indication that Congress considered, much less intended, software to be included under the second definition of work made for hire.[220] An interpretation of § 101 that can overcome these obstacles can potentially be a viable solution to the problems posed by software terminability.

C. Will Large-Scale Computer Software Programs Be Terminable?

Another important factor in determining the scope of software terminability is whether large-scale, multiauthor software works will be terminable. Copyrightable works with more than one author can classify as either "joint works" or "collective works" under the Copyright Act.[221] Joint works are prepared by multiple authors who share the intent to merge their respective individual contributions into a unitary and inseparable work.[222] An example of a joint work is a book coauthored by two individuals, with the copyright vesting in both authors as co-owners.[223] Collective works also require multiple authors who each contribute individually copyrightable works that are assembled into a whole.[224] However, collective works are distinct from joint works in two crucial ways.[225] First, collective works do not require the authors to share the intent to merge the works.[226] Second, although the copyright in the collective work vests in its creator, the contributing authors each retain a copyright in their own contribution to the collective work, making the final work separable as a result.[227]

Large-scale software works could potentially fall within either category given that they share qualities of both joint works and collective works.[228] In order to function, the finished software program must run as a complete and cohesive unit, which appears analogous to a joint work such as a co-produced movie.[229] On the other hand, programs can be deconstructed into their individual building blocks of code, which instead seems analogous to a collective work, such as an encyclopedia.[230] Although the Copyright Act defines and distinguishes the two categories,[231] predicting how courts will apply them to computer software requires looking at relevant case law.

219. *See* Armstrong, *supra* note 13, at 422–23.

220. *See generally* H.R. REP. NO. 94-1476 (1976) (containing no discussion or reference to software's potential inclusion under § 101(2)).

221. *See* Siniouguine v. Mediachase Ltd., No. CV 11-6113, 2012 U.S. Dist. LEXIS 87190, at *5 (C.D. Cal. June 11, 2012); Harris, *supra* note 71, at 692–93.

222. 17 U.S.C. § 101 (defining "joint work").

223. H.R. REP. NO. 94-1476, at 121 (explaining how co-owners of copyright in a joint work are to be treated as tenants in common).

224. 17 U.S.C. § 101 (defining "collective work").

225. H.R. REP. NO. 94-1476, at 121 (distinguishing collective works from joint works).

226. *Id.*

227. *Id.*

228. *See* Siniouguine v. Mediachase Ltd., No. CV 11-6113, 2012 U.S. Dist. LEXIS 87190, at *5 (C.D. Cal. June 11, 2012); Harris, *supra* note 71, at 692–93.

229. *See* Harris, *supra* note 71, at 694–95.

230. *Id.* at 690 n.159.

231. 17 U.S.C. § 101.

The Ninth Circuit addressed issues pertaining to joint works in *Aalmuhammed v. Lee*,[232] when a substantial contributor to a movie brought suit in an effort to establish the movie as a joint work of authorship in which he is entitled co-ownership.[233] The court's analysis focused on the parties' lack of shared mutual intent for the rights to the movie to vest in both the director and the contributing plaintiff as co-authors.[234] In addition, the court found the most important factor to be the exercise of control over the work.[235] More specifically, the court found that the author is likely to be "the inventive or master mind" who "creates, or gives effect to the idea."[236] In the context of a film production, the court determined that the defendants, as producer and director, exercised sufficient creative control to be considered the sole authors.[237] While the court did recognize that the plaintiff's contribution would have been copyrightable in isolation, its relative importance in the context of the movie as a whole was insufficient to warrant joint authorship, and thus, co-ownership in the movie's copyright was denied.[238]

The court's holding in *Aalmuhammed*—that a substantial creative contribution is necessary but not sufficient to establish joint authorship[239]—is relevant to copyrights held in large-scale computer software. From a public policy perspective, the court argued in a later case that providing every single contributor to complex works with a copyright interest would "make Swiss cheese of copyrights."[240] Furthermore, the risks associated with multitudinous claims of copyright in complex works naturally grow with the number of contributing parties.[241] The Ninth Circuit has not been alone in its determination, as the Second Circuit has similarly stated that "[w]e agree with the *en banc* Ninth Circuit . . . that the creation of 'thousands of standalone copyrights' in a given work was likely not intended."[242] The above considerations are particularly relevant to the software industry due to the rapid growth in both the volume and complexity of big code.[243] Courts' opposition to creating excessive amounts of standalone copyrights in a given work thus appears to foreclose the classification of large-scale works as collective works.

If programmers who contributed to big-code works attempt to terminate the assignment of said works, courts would be faced with a similar analysis

232. 202 F.3d 1227 (9th Cir. 2000).
233. *Id.* at 1230.
234. *Id.* at 1234–35.
235. *Id.* at 1234.
236. *Id.*
237. *Id.* at 1236.
238. *Id.* at 1233–34.
239. *See id.* at 1233.
240. Garcia v. Google, Inc., 786 F.3d 733, 742 (9th Cir. 2015).
241. *See Aalmuhammed*, 202 F.3d at 1232 ("But as the number of contributors grows and the work itself becomes less the product of one or two individuals who create it without much help, the word is harder to apply.").
242. Casa Duse, LLC v. Merkin, 791 F.3d 247, 259 (2d Cir. 2015) (quoting *Garcia*, 786 F.3d at 743).
243. *See* SOURCEGRAPH, *supra* note 151, at 2 (demonstrating through survey data that big code has grown substantially in volume, variety, velocity, and value).

as the Ninth Circuit was in *Aalmuhammed*.[244] Just as writers, producers, actors, and stage crews contribute to a movie, project sponsors, project managers, software developers, and testers are only a few of the different roles contributing to a complete software project.[245] While each of the parties involved may have made a copyrightable contribution to the overall work, it is insufficient for copyright to vest if the parties were not also exercising control over the project as the "master mind."[246] This is true even if the § 101 intent requirement of joint works is satisfied.[247] Thus, following *Aalmuhammed*, courts also appear unlikely to classify large-scale software works as joint works because no individual programmer would satisfy the "master mind" standard.[248]

The standard set in *Aalmuhammed* may be best understood as a de facto extension of the work made for hire exception through common law.[249] This is because, in large-scale projects involving many individually copyrightable contributions, ownership of the work may vest in the hiring party even absent an employment relationship or work made for hire agreement.[250] By analogizing large-scale software works to other multiauthor works, such as movies, courts can successfully avoid threats of termination that could otherwise jeopardize the big-code sector of the proprietary software market.

D. Will § 117 of the Copyright Act Mitigate the Effects of Software Terminability?

Due to the unique aspects of computer programs, which distinguish them from other copyrightable works, Congress established CONTU to study issues arising from software copyrightability.[251] CONTU summarized its findings and recommendations in a final report to Congress.[252] Congress acted on the report's recommendations, most notably by adding § 117 to the Copyright Act and amending § 101 to include a statutory definition of "computer program."[253] The CONTU report explains the justifications for adding § 117, namely, to immunize users of computer programs from infringement suits that could potentially arise from otherwise innocuous behavior.[254] For example, running a software disk on a computer results in

244. *See* Harris, *supra* note 71, at 667–68.
245. *See* Simon Swords, *Software Development Project Roles and Responsibilities*, ATLAS COMPUT. SYS. LTD. (Nov. 21, 2017), https://www.atlascode.com/blog/software-development-project-roles-and-responsibilities/#PROJECT_SPONSOR [https://perma.cc/ULT8-MVHS].
246. *See Aalmuhammed*, 202 F.3d at 1232–33.
247. *Id.* at 1233.
248. *See id.* at 1232.
249. *See id.* at 1235–36.
250. *Id.* at 1236.
251. Act of Dec. 31, 1974, Pub. L. No. 93-573, 88 Stat. 201.
252. *See* CONTU Report, *supra* note 91, at 1213.
253. 17 U.S.C. § 101 ("A 'computer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."); *see* Natalie Heineman, Note, *Computer Software Derivative Works: The Calm Before the Storm*, 8 J. HIGH TECH. L. 235, 239 (2008).
254. *See* CONTU Report, *supra* note 91, at 12–13.

making a nominal "copy" of the program that is displayed on the computer screen, which would constitute infringement under the plain text of the Copyright Act.[255] But CONTU posited that it makes little sense to expose every end user of software to an infringement suit for taking a necessary step in the utilization of their rightfully owned software.[256]

Congress rectified the issue of infringing copies by creating a safe harbor under § 117(a) for "a new copy or adaptation [that] is created as an essential step in the utilization of the computer program in conjunction with a machine and . . . is used in no other manner . . . ."[257] This safe harbor has been interpreted by courts as protecting necessary copies through the language "new copy,"[258] as well as minor alterations that are necessary to use the program through the language "adaptation."[259] In effect, the safe harbor extends to cover minor bug fixes and alterations to source code, as long as they are necessary for the use of the software for its intended purpose.[260] Therefore, the § 117(a) safe harbor is beneficial for owners of internal software systems, such as the aforementioned COBOL systems, because necessary minor changes to source code are immunized from infringement claims.

Alterations and bug fixes, however, remain a substantial issue for OSS works.[261] Although necessary adaptations are permissible, the lease, sale, or transfer of those particular adaptations are still prohibited, unless authorized by the copyright owner under § 117(b).[262] If an author of an OSS work were to terminate a license, individuals would still be able to use the program; however, it would essentially be "frozen" in place.[263] Users would no longer be able to share and receive improvements to the program with the community, thus losing one of the open source model's primary benefits.[264] In theory, users would be able to share their improvements if they receive permission from the original author, as is permitted under the statute.[265] However, in practice, this provision is irrelevant because authors or heirs who terminate their licenses would have no incentive to then allow continued sharing on an individual basis.[266] Much like the Copyright Act as a whole, Congress presupposed a proprietary incentive structure when drafting § 117, which is now coming into conflict with the nonproprietary interests of OSS.[267] Therefore, the necessary solution for incorporating OSS works into the Copyright Act is a legislative amendment.

255. *See id.*
256. *Id.*
257. 17 U.S.C. § 117(a)(1).
258. *See, e.g.*, Krause v. Titleserv, Inc., 402 F.3d 119, 122 (2d Cir. 2005).
259. *See id.* at 125–26.
260. *See* Phelps, *supra* note 19, at 270.
261. *Id.* at 271.
262. *Id.* at 271–72.
263. *Id.* at 271.
264. *Id.*
265. *See* 17 U.S.C. § 117(b).
266. *See* Phelps, *supra* note 19, at 271–72.
267. *See* Armstrong, *supra* note 13, at 416.

III. Preventing the Inequitable Effects of Software Terminability

Resolving the impending issues that will arise from software terminability requires a nuanced solution that addresses both proprietary and open source sectors of the software industry. A viable solution must address the specific problems at issue in software without making changes that extend beyond the scope of the problem and cause unintended effects to other copyrightable works. For this reason, it is prudent to begin with the Copyright Act in its current form to determine which problems can be dealt with sufficiently through the law as it currently stands and which others cannot and thus require further legislative action.

This Note proposes a two-pronged solution that addresses both of the above considerations. First, this Note advocates for a judicial solution to termination issues arising in proprietary software. This solution posits that, through statutory interpretation of the Copyright Act, judges can reasonably expand the scope of work made for hire, and as a result, narrow the scope of terminable works. Second, this Note proposes a legislative amendment to the Copyright Act in an effort to protect OSS works from termination. Unlike proprietary software, the protection of OSS requires a legislative amendment because the limitations of works made for hire, large-scale works, and § 117 fail to adequately protect OSS as a nonproprietary work that was not originally envisioned by Congress when drafting the Copyright Act.

A. Proposed Statutory Interpretation of the Copyright Act

To address issues arising in proprietary software, courts can mitigate the most harmful effects of termination by employing a series of limiting factors. These limiting factors consist of works made for hire, common law exemptions of large-scale works, and § 117 of the Copyright Act.

The work made for hire provision is the most important limiting factor due to its ability to circumvent the termination right's inalienability.[268] Adoption of a novel interpretation of work made for hire under § 101(2) will expand the scope of the limitation and reduce the threat of termination as a result. The primary issue with § 101(2) is that computer software does not explicitly appear as one of the nine enumerated categories of works that can be commissioned as works made for hire.[269] Some district courts have already begun to fit computer software within these categories as a "compilation," "contribution to a collective work," or "audiovisual work."[270] None of the circuit courts have adopted any of the foregoing interpretations, yet said interpretations have not been categorically rejected either.[271] With the current status of software's placement within the § 101(2) categories in

268. NIMMER & NIMMER, *supra* note 217, § 11.02[A][2].

269. *See* 17 U.S.C. § 101(2).

270. *See supra* notes 213–15215 and accompanying text.

271. Ben Bhandhusavee, *Is Your Customized Software a 'Work for Hire'?*, BHANDLAW (Dec. 31, 2019), https://www.bhandlaw.com/2019/12/31/is-custom-software-a-work-made-for-hire/ [https://perma.cc/T6BZ-BX2D].

limbo, an interpretation with a more robust textual basis could successfully convince the circuit courts on appeal.

This Note proposes that interpreting computer programs as "instructional text[s]" pursuant to the sixth category listed under § 101(2) can accomplish this goal.[272] Unlike the other eight categories, there is a statutory definition for "instructional text" within § 101(2) that provides a strong basis for including computer programs within the statute.[273] To begin with, § 101 defines "computer program" as "a set of statements or *instructions* to be used directly or indirectly in a computer in order to bring about a certain result."[274] The key term "instructions" appears to align neatly with an "instructional text" as required by § 101(2).[275] The statutory definition of "instructional text" is "a literary, pictorial, or graphic work prepared for publication and with the purpose of use in systematic instructional activities."[276] Once again, a similar issue arises in that computer programs are not explicitly listed as an instructional text, as they are likewise absent from the nine enumerated categories of § 101(2). Yet, what is included in the definition of "instructional text" is "a *literary* . . . work."[277] Given that courts have consistently interpreted computer programs as "literary works" pursuant to § 102 in establishing their copyrightability,[278] a consistent usage of statutory language also permits their inclusion as a work made for hire under § 101(2).

This proposed course of statutory interpretation is novel as applied to the words at issue but is representative of two widely recognized canons of statutory interpretation known as the "whole act rule" and the "presumption of consistent usage."[279] The whole act rule is employed by courts to interpret individual words or phrases in the context of the entire act.[280] As applied here, the second definition of work made for hire is being interpreted in light of two other definitions included under § 101. The presumption of consistent usage is self-explanatory in that the court will presume that words that are repeated throughout the act will bear the same meaning as previously and subsequently used.[281] As applied here, the presumption of consistent usage supplements the whole act rule analysis by presuming that the repetition of both "instruction" and "literary work" bear the same meaning throughout the Copyright Act.

The effect of courts' adoption of the proposed interpretation would be substantial. Recognizing that computer programs can be contracted into as a

272. 17 U.S.C. § 101(2).
273. *Id.*
274. *Id.* (emphasis added).
275. *Id.*
276. *Id.*
277. *Id.* (emphasis added).
278. *See* Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1249 (3d Cir. 1983).
279. *See* Abbe R. Gluck & Lisa Schultz Bressman, *Statutory Interpretation from the Inside—An Empirical Study of Congressional Drafting, Delegation, and the Canons: Part I*, 65 STAN. L. REV. 901, 937 (2013).
280. *See id.* at 930.
281. *Id.* at 937 n.106.

work made for hire will essentially circumvent the inalienability of the termination right.[282] The benefits of this solution are that the courts need not wait for Congress to amend the Copyright Act and that the interpretation is specifically tailored to computer software, thus avoiding unintended consequences on other works that may result from a legislative amendment.

Those who may challenge the above method of interpretation would likely point to the absence of congressional intent indicating that the second definition of work made for hire was intended to encompass computer programs. There is merit to this challenge, given that at least one circuit court has interpreted § 101 as indicative of congressional intent to create both a broad catchall through the first definition and narrow carveout through the second definition.[283] However, the Supreme Court has long been careful not to interpret the absence of congressional intent as evidence of congressional disapproval.[284] Thus, while Congress may not have intended to include software within § 101(2) when enacting the Copyright Act, the language of "instructional text" is sufficiently broad to encompass computer programs in light of their statutory definition.

Another critique of this approach is that it may appear inequitable that programmers can have their termination right circumvented retroactively through a novel interpretation of § 101(2). Yet, concerns of this nature are mitigated by the other two requirements of § 101(2): (1) that the work is specially commissioned and (2) that it is agreed to constitute a work made for hire through a signed contract.[285] Programmers who voluntarily agreed to sign away their works as works made for hire cannot reasonably claim that they expected the copyright to nonetheless vest in themselves as the author. Perhaps programmers who paid careful attention to the enumerated categories of § 101(2) can plausibly claim as much—yet that would have required them to sign a contract they believed to be legally unsound, thus violating the implied duty of good faith and fair dealing inherent in contractual agreements[286] and vitiating any claim of inequity as a result. Therefore, retroactive application of the novel interpretation of § 101(2) is a sufficiently equitable result for both contracting parties regardless of their knowledge of the statute or lack thereof.

282. NIMMER & NIMMER, *supra* note 217, § 11.02[A][2].

283. *See* Easter Seal Soc'y for Crippled Child. & Adults of La., Inc. v. Playboy Enters., 815 F.2d 323, 331 (5th Cir. 1987) ("These categories are accorded special treatment, and the buyer will be the author only if he has complied with the requirement of a written agreement. In other words, § 101(2) carves out special protections from the expansive old doctrine for a narrow group of sellers.").

284. *See* Girouard v. United States, 328 U.S. 61, 69 (1946) ("It is at best treacherous to find in Congressional silence alone the adoption of a controlling rule of law."). However, the U.S. Supreme Court has recently engaged in a similar form of statutory interpretation in which the Court interpreted the word "sex" in light of its plain meaning to include sexual orientation, despite the absence of congressional intent at the time of enactment. *See* Bostock v. Clayton Cnty., 140 S. Ct. 1731, 1737 (2020).

285. 17 U.S.C. § 101(2).

286. *See* U.C.C. § 1–304 (AM. L. INST. & NAT'L CONF. OF COMM'RS ON UNIF. STATE L. 2021).

In addition to retroactivity concerns, prospective concerns could be raised about the threat of unremunerative transfers.[287] However, programmers are better suited to overcome this imbalance and negotiate around § 101(2) due to the distinguishing factors that set them apart from most other authors who require such protection.[288] Given that works made for hire under § 101(2) must be specially commissioned, it is reasonable to infer that purchasers who reach out to programmers are doing so because they require a particular service or expertise. Programmers have far more negotiating power in this circumstance as opposed to authors of other works because both parties have a mutual need.[289] Therefore, programmers need not sign contracts that specify that their works are to be made for hire.[290] Programmers can weigh the short-term and long-term benefits of either foregoing or retaining their termination right in the process of negotiating the terms of contracts. Overall, the resulting dynamic of the proposed interpretative solution provides more transparency in the negotiation process, while also providing sufficient flexibility for programmers to leverage their termination rights.

B. Legislative Amendment to Solve Termination Issues in Open Source Software

Unlike proprietary software, OSS requires a legislative solution because the limiting factors on termination contained in the Copyright Act are largely inapplicable to OSS. The work made for hire doctrine, as the most important limitation, is largely absent from OSS because contributions are made on a voluntary basis rather than through an employment relationship.[291] Similarly, the de facto exception for large-scale works is essentially a nonfactor because the use of licensing agreements in OSS precludes satisfaction of the necessary intent-to-merge requirement.[292] Lastly, § 117 fails to protect a vital function of OSS, in that distribution of derivative works, including bug fixes and updates, constitutes infringement and is thus prohibited under the Copyright Act.[293] The above shortcomings are not rectifiable through judicial interpretation of the Copyright Act in its current form, thus making a legislative amendment the necessary solution.

287. *See supra* notes 173–17878 and accompanying text.

288. *See supra* notes 173–17878 and accompanying text.

289. *See supra* notes 171–78 and accompanying text.

290. This is especially true of proficient and experienced programmers, as expressed by a quote attributed to Bill Gates: "A great lathe operator commands several times the wages of an average lathe operator, but a great writer of software code is worth 10,000 times the price of an average software writer." *See* Reed Hastings, *Netflix CEO on Paying Sky-High Salaries: 'The Best Are Easily 10 Times Better Than Average,'* CNBC (Sept. 8, 2020, 10:02 AM), https://www.cnbc.com/2020/09/08/netflix-ceo-reed-hastings-on-high-salaries-the-best-are-easily-10x-better-than-average.html [https://perma.cc/8UGX-H4XR].

291. *See* ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY 57 (Tim O'Reilly ed., 2d ed. 2001) ("[O]pen-source developers are volunteers, self-selected for both interest and ability to contribute to the projects they work on . . . ."). *But see* Armstrong, *supra* note 13, at 405 n.273 (highlighting the adoption and growth of OSS development in some private companies).

292. *See supra* note 222 and accompanying text.

293. *See supra* Part II.D.

One scholar has advocated for a legislative amendment which would allow copyright owners to abandon their copyrights by making an express dedication of their works to the public domain.[294] This solution is particularly attractive because it draws from an existing provision of the Patent Act of 1952, which enables patent abandonment, to create an analogous right in the Copyright Act.[295] The public policy effects of patent abandonment are assessable by legislators, thus enabling them to draw reasonable conclusions about how abandonment would likely play out in copyright.

However, the proposed amendment is deficient because, while the original OSS program is placed in the public domain through abandonment, all subsequent derivative works are free from licensing constraints and may, in essence, cease to be OSS works. This is because the original author of the program cannot retain perpetual open access to source code through licensing arrangements such as the GPL once the author's program is abandoned. Any additional code added by a subsequent author can be kept secret from the public, thus making the closed portion of the derivative work's code indistinguishable from other proprietary software. Given that perpetual open access to code is one of the core features of OSS, its loss makes this solution less than ideal.

Instead, Congress should enact legislation that creates a voluntary compulsory licensing scheme for OSS works. Compulsory licensing schemes already exist within the Copyright Act, though they are mostly limited to the music industry.[296] Under § 115, artists are required to license the underlying musical compositions of their songs to others who wish to use the copyrights for permissible purposes, such as creating a cover of said songs.[297] This provision was intended to strike a balance between the original artists' rights and subsequent artists' ability to draw from a well of existing creativity in the furtherance of their own creative efforts.[298] In other words, the compulsory licensing scheme shares many of the same justifications underlying the OSS movement.[299] Thus, § 115 serves as a natural starting point in creating a legislative solution for OSS.

Congress should recognize and define OSS by adding a statutory definition of "open source software" under § 101 of the Copyright Act. Next, a centralized licensing body should be created to act as the intermediary

294. *See* Armstrong, *supra* note 13, at 419.

295. *Id.*

296. *See* Paul S. Rosenlund, Note, *Compulsory Licensing of Musical Compositions for Phonorecords Under the Copyright Act of 1976*, 30 HASTINGS L.J. 683, 694 (1979) (noting that, under the Copyright Act, most other types of copyrightable works are not subject to the same compulsory licensing requirements as music).

297. *See* 17 U.S.C § 115(a)(2). However, the license is limited insofar as the licensee cannot "change the basic melody or fundamental character of the work" and will not obtain any rights in their cover as a derivative work without first receiving the copyright owner's consent. *Id.*

298. *See* Jacob Victor, *Reconceptualizing Compulsory Copyright Licenses*, 72 STAN. L. REV. 915, 952 (2020).

299. *See supra* notes 21–22 and accompanying text.

between the original authors of OSS programs and those who wish to use or alter the program. In practice, the licensing body would function similarly to the Mechanical Licensing Collective, a nonprofit organization that is responsible for administering compulsory licenses in the music industry.[300] That is, the licensing body would create a database of OSS works,[301] enforce either the default "blanket license"[302] or a different license upon which both parties have agreed,[303] and oversee the payment of royalties.[304] The licensing body would not be responsible for regulating the substance of the agreement, as long as it is consistent with the statutory definition of an OSS work. Most OSS works today are licensed through standardized agreements such as the GPL, effectively minimizing transaction costs of the case-by-case negotiation of terms.[305] The licensing body would facilitate this process by establishing the GPL as the default license to which the licensee and licensor reserve the right to opt out of for a privately negotiated license.

The benefits of the centralized nature of the licensing body come into play when considering derivative works. The licensing body will have on file the information of all users of the original OSS program, such that a programmer seeking to license a derivative of the original can do so through the same process. The author of the derivative work could use either the default licensing agreement, or one that is less restrictive, but could not use a more restrictive agreement. Thus, perpetual open access to the original program's source code is not only guaranteed under this proposal just as it is currently but also made more accessible and accountable through the centralized licensing body.

Finally, to rectify the issues posed by termination, Congress should add a provision to § 203 specifying that OSS works issued through a compulsory licensing system cannot be terminated. This provision is necessary to ensure that the OSS works remain accessible to the public in perpetuity. Concerns regarding unremunerative transfers are moot under a compulsory licensing system, given that they operate based on a fixed royalty rate from which the licensors automatically receive payment upon use of their works.[306] Absent the core justification underlying the existence of the termination right, there is little reason to retain termination in OSS works under a compulsory licensing system. The one caveat is that transfers of the original work itself should remain terminable, as such transfers carry with them the royalties earned through compulsory licenses. Thus, while public access through compulsory licensing remains open in perpetuity regardless of who owns the original, the original author of the OSS work is free to maintain ownership,

300. *See Musical Works Modernization Act*, U.S. COPYRIGHT OFF., https://www.copyright.gov/music-modernization/115/ [https://perma.cc/NP3S-Y8TE] (last visited Oct. 29, 2021).
301. *See* 17 U.S.C. § 115(d)(3)(E).
302. *Id.* § 115(d)(1)(B).
303. *Id.* § 115(d)(1)(C).
304. *Id.* § 115(d)(3)(G).
305. *See* Phelps, *supra* note 19, at 263.
306. 17 U.S.C. § 115(d)(8).

or if the author chooses to transfer it, the author may recover ownership through termination thirty-five years later. This arrangement would be ideal, as it would strike the balance between public access and private ownership while avoiding the issue of unremunerative transfers altogether.

Although a compulsory licensing system may be a viable solution to the problems posed to OSS by termination, the intricacies of a full amendment to the Copyright Act will take a substantial amount of time to explore. This is evident from the music industry, which has seen multiple revisions of its compulsory licensing system, the most recent of which is the Orrin G. Hatch-Bob Goodlatte Music Modernization Act.[307] It would be prudent to wait and see how recent changes to the compulsory licensing framework play out in practice before implementing an analogous system for OSS. Fortunately for Congress, the gap in time before termination issues arise is larger for OSS, as it only started to become "mainstream"[308] in the early 2000s. Due to the thirty-five-year threshold, as well as the time it may take for OSS authors to both recognize that they have a termination right and then proceed to exercise it, Congress may have well over a decade to prepare the proposed amendment.

In the meantime, perhaps it would be worthwhile for Congress to take the first step, as advocated by the Second Circuit nearly three decades ago, by commissioning a "CONTU II" to consider updates to the Copyright Act that reflect the rapidly evolving software industry.[309] At the forefront of such considerations should be solutions to the intersection of termination and software. The issues raised and the solutions presented by this Note can contribute to this ongoing discussion.

CONCLUSION

Technological advancement and the law are rarely, if ever, working in tandem. By including computer software as copyrightable subject matter, Congress and the courts alike have treated software much the same as art, movies, and literature. However, due to the rapid growth of the software industry and the unforeseen development of OSS, it is time for Congress to recognize the unique threat that copyright termination poses to software in particular. The fact that software termination may be years away is not an excuse for complacency. Rather, Congress should view it as an opportunity to preemptively rectify the law before its economic consequences are felt.

The solutions presented by this Note address proprietary software and OSS with the respective levels of urgency they require. Courts can adopt the proposed interpretation of "instructional texts," such that proprietary software may be considered work made for hire, thus making what was

307. Pub. L. No. 115-264, 132 Stat. 3676 (2018) (codified as amended in scattered sections of 17, 19, and 28 U.S.C.).

308. Steve Lohr, *Code Name: Mainstream; Can 'Open Source' Bridge the Software Gap?*, N.Y. TIMES (Aug. 28, 2000), https://www.nytimes.com/2000/08/28/business/code-name-mainstream-can-open-source-bridge-the-software-gap.html [https://perma.cc/823C-QH34].

309. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992).

previously an unalienable termination right a de facto alienable right. A judicial solution such as this is ideal for proprietary software because the thirty-five-year termination threshold will be reached much sooner than the window for OSS. The extended gap of time before OSS reaches the threshold provides Congress with ample opportunity to consider and draft a solution. Accordingly, this Note's legislative proposal for a compulsory licensing system is tailored to meet Congress's time frame.

Issues raised by software terminability represent a larger tendency of the law to lag behind technological development, which is an unfortunate, but not unavoidable, phenomenon. By preemptively addressing future problems, the law may finally start to facilitate, rather than hinder, the inevitability of technological progression.