The Uneasy Case for Software Copyrights Revisited

*Pamela Samuelson**

INTRODUCTION

Forty years ago, Justice Stephen Breyer expressed serious doubts about the economic soundness of extending copyright protection to computer programs in his seminal article, *The Uneasy Case for Copyright*.[1] A decade later, Congress enacted legislation to protect programs through copyright law, notwithstanding Breyer's cogently expressed doubts.[2] This Article revisits *The Uneasy Case* to consider whether Breyer's skepticism about copyright for computer programs was warranted at the time, as well as whether the case for copyrighting computer programs has become easier over time. As to the first question, the answer is yes; Breyer's skepticism was warranted at the time.[3]

---

* Richard M. Sherman Distinguished Professor of Law, Berkeley Law School. I wish to thank Bob Brauneis and his colleagues at the George Washington University Law School for the invitation to participate in the Symposium as well as Micah Gruber, Kathryn Hashimoto, and Andrea Yankovsky for research support for this article.

1 Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281 (1970).

2 *See* Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3015 (codified at 17 U.S.C. §§ 101, 117). Although some claim that Congress extended copyright protection to computer programs when it enacted the Copyright Act of 1976, there is some ambiguity in the legislative history on this point. *Compare* NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 15–16 (1979) [hereinafter CONTU REPORT], *with* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Executable Form*, 1984 DUKE L.J. 663, 663 [hereinafter Samuelson, *CONTU Revisited*]. However, any such ambiguity was resolved by the 1980 amendments. *See* Act of Dec. 12, 1980, 94 Stat. 3015. Breyer's skepticism about software copyrights would have been familiar to Professor Arthur R. Miller, who chaired the National Commission on New Technological Uses of Copyrighted Works ("CONTU") subcommittee that recommended copyright as a form of legal protection for computer programs. *See* Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEX. L. REV. 1921, 1949–50 & n.191 (2007) [hereinafter Samuelson, *Why Copyright Excludes*]. Miller was not only Breyer's colleague at Harvard Law School, *id.*, but also someone who had once shared Breyer's skepticism about copyright as a form of intellectual property protection for computer programs, *see Copyright Law Revision: Hearing on S. 597 Before the Subcomm. on Patents, Trademarks, and Copyright of the S. Comm. on the Judiciary*, 90th Cong. 196–97 (1967) (statement of Arthur R. Miller, Chairman, Nat'l Comm'n on New Technological Uses of Copyrighted Works) (characterizing computer programs as "functional item[s]" that were quite distinct from "books or plays or motion pictures or poetry—the forms of expression that traditionally have been covered by our copyright legislation," and expressing concern about applying copyright to programs).

3 *See infra* Part I.

As to the second question, the answer is also yes; the case for copyrighting computer programs did become easier over time.[4]

As Breyer observed, the mere fact that computer programs are expensive to develop and cheap to copy does not mean that copyright protection should be available for them.[5] The more important issue is whether program developers are able to recoup the costs of development in a meaningful way; copyright protection may not be necessary to achieve this goal. Breyer was astute in his empirical approach to assessing the state of this industry in considering how it bore on the economic argument for copyright protection. Breyer showed that, in 1970, there were numerous ways that firms recouped investments in software; hence, the state of the industry at that time provided scant support for copyrighting computer programs.[6] Yet Breyer was remarkably prescient in articulating a set of economic indicators that—should they occur (and, in fact, they did)—would strengthen the case for copyright for computer programs.[7]

The spectacular growth of the computer software industry in the United States since 1970 has seemingly vindicated the wisdom of Congress's decision to protect software by copyright law.[8] Indeed, copyright has become an international norm as a legal means of protecting software.[9] Even so, certain developments in the modern era present a somewhat murkier and more complicated picture of the case for copyright protection for computer programs.[10] Who, for instance, really needs a copyright if all commercially significant computer programs

---

4 *See infra* Parts II–III.

5 Breyer, *supra* note 1, at 344. Under this rationale, Congress might well have extended copyright to protect the layout of semiconductor-chip circuitry, as indeed Professor Miller recommended as a logical move. *See* Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 1045 (1993). Yet, Congress decided not to do this because of the functionality of chip circuitry designs. *See, e.g.*, Pamela Samuelson, *Creating a New Kind of Intellectual Property Law: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471, 473–74 (1985).

6 Breyer, *supra* note 1, at 344–50. Breyer's analysis is discussed in Part I. The CONTU Report is, by contrast, almost devoid of empirical information about the state of the software industry or of careful economic analysis about the implications of extending copyright protection to computer programs. *See* CONTU REPORT, *supra* note 2, ch. 3.

7 Breyer, *supra* note 1, at 347. These factors are discussed in detail *infra* notes 57–58 and accompanying text.

8 *See infra* Part II.

9 *See* Agreement on Trade-Related Aspects of Intellectual Property Rights art. 10(1), Apr. 15, 1994, 33 I.L.M. 1197, 1869 U.N.T.S. 300 ("Computer programs, whether in source or object code, shall be protected as literary works under the Berne Convention (1971).").

10 *See infra* Part III.

migrate to "the cloud"?[11] This Article discusses these developments and explores how they fit into Breyer's economic analysis of copyright protection.

Part I of this Article describes the state of the software industry at the time Justice Breyer published *The Uneasy Case*, and explains why he was skeptical about extending copyright protection to computer software. Part II describes the phenomenal growth of the software industry during the thirty years after *The Uneasy Case* was published, and discusses how copyright protection for computer software may have facilitated that growth. Finally, Part III suggests that the case for copyright might have weakened in the twenty-first century due to emerging technologies and trends.

I. THE UNEASY CASE FOR SOFTWARE COPYRIGHTS IN THE 1960s–1970s

A. *The Software Protection Debate Leading Up to* The Uneasy Case

There was considerable debate in the 1960s, during the gestation of the legislation that became the Copyright Act of 1976,[12] about whether computer programs could, or should, be protected by copyright law.[13] Although no one seriously questioned that source code forms of programs could be copyrighted as written texts,[14] there were two principal concerns about applying copyright to machine-executable forms of programs: first, executable forms of programs are functional processes, a class of intellectual creation that has traditionally been ineligible for copyright protection;[15] and second, machine code

---

11 *See, e.g.*, DELOITTE, CLOUD COMPUTING: FORECASTING CHANGE 5–7 (2009), *available at* http://www.deloitte.com/assets/Dcom-Ireland/Local%20Assets/Documents/ie_Consulting_CloudComputing_09.pdf; *see also infra* notes 250–54 and accompanying text.

12 Copyright Act of 1976, Pub. L. No. 94-553, § 101, 90 Stat. 2541 (1976) (codified as amended in sections 17 U.S.C. §§ 101–1332).

13 The legislative history is discussed at length in Samuelson, *Why Copyright Excludes*, *supra* note 2, at 1945–51.

14 *See, e.g.*, Breyer, *supra* note 1, at 340 n.233 (noting that the copyright subject matter provision in the proposed revision of copyright law (later codified as 17 U.S.C. § 102(a) (2006)) seemed broad enough to encompass computer programs).

15 The principle that copyright law does not protect functional creations has longstanding support in American copyright law. *See, e.g.*, Baker v. Selden, 101 U.S. 99, 103, 107 (1879) (holding that copyright protection does not extend to useful arts, such as bookkeeping systems, that are depicted in copyrighted works). For a detailed analysis of the *Baker* rule and its implications on copyright as applied to computer programs, see Samuelson, *Why Copyright Excludes*, *supra* note 2, at 1961–73. Computer programs are best understood as machines for which the medium of construction just happens to be text. *See* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2320–24 (1994).

was arguably not a "copy" of the source code within the meaning of copyright law, in part because it was not intended for communication with humans.[16] Despite these two concerns, some commentators argued that copyright should be extended to computer programs.[17]

Indeed, the U.S. Copyright Office ("Office") began issuing registration certificates for computer programs in the mid-1960s.[18] But these certificates were issued under the Office's "rule of doubt."[19] That is, although the Office was willing to issue certificates to program developers who were prepared to argue in court that the registered programs were in fact copyrightable, the certificates reflected the Office's doubts about whether executable forms of computer programs really qualified for copyright protection.[20] Because of these doubts, some commentators suggested that a sui generis form of legal protection for computer programs might be a better alternative than copyright.[21]

---

16 Breyer, *supra* note 1, at 340 n.233 (noting that some doubts existed about whether computer programs were "original works of authorship" in a constitutional sense given that they are not literary or artistic in content, and do not convey information to readers); *see also* White-Smith Music Publ'g Co. v. Apollo Co., 209 U.S. 1, 18 (1908) (ruling that piano rolls were not "copies" of copyrighted musical compositions in part because the rolls could not be read by humans). One early software copyright case used the following analogy to explain why executable forms of programs were not copyrightable: Source code is akin to architectural plans, both of which could be copyrighted. By contrast, object code is analogous to a house built with the copyrighted architectural plans, but neither—in that court's judgment—qualify for copyright protection under then-existing U.S. law. *See* Data Cash Sys., Inc. v. JS&A Grp., Inc., 480 F. Supp. 1063, 1068 (N.D. Ill. 1979), *aff'd on other grounds*, 628 F.2d 1038 (7th Cir. 1980).

17 *See* Breyer, *supra* note 1, at 343, 343 n.244 (citing sources that supported copyright protection for computer programs).

18 *See* U.S. COPYRIGHT OFFICE, LIBRARY OF CONG., CIRCULAR 31D (1965) [hereinafter CIRCULAR 31D], *reprinted in* Duncan M. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611, 652 n.72 (1983).

19 *Id*.

20 *See id.* In general, "[t]he registerability of computer programs involves two basic questions: (1) whether the program is . . . a 'writing of an author' and thus copyrightable, and (2) whether a reproduction of the program in a form actually used to operate or to be 'read' by a machine is a 'copy' and can be accepted for copyright registration." *Id.* Both were "doubtful questions," but the Register nevertheless decided to accept programs for registration as long as the program was published with proper copyright notices, and the full source code was deposited with the Office. *Id*.

21 *See, e.g.*, Elmer Galbi, *Proposal for New Legislation to Protect Computer Programming*, 17 BULL. COPYRIGHT SOC'Y 280, 283–92 (1970). The World Intellectual Property Organization promulgated a sui generis proposal in the 1970s. *See* WORLD INTELLECTUAL PROP. ORG., INT'L BUREAU, MODEL PROVISIONS ON THE PROTECTION OF COMPUTER SOFTWARE (1978). Much of CONTU's argument in favor of copyright for programs was based on an assessment that it was a better fit than other existing legal protection. CONTU REPORT, *supra* note 2, at 17–18. CONTU did not consider a sui generis option. *Id.*

Breyer's skepticism about the case for software copyrights was bolstered by the fact that only about 200 computer programs had actually been registered with the Office in the five years after the Office announced its willingness to issue registrations.[22] This suggested that software developers did not consider copyright to be essential to their businesses.[23] Moreover, it appeared that, in the 1960s, the software industry was "burgeoning without the use of copyright."[24] Breyer pointed to an electronic data-processing industry newsletter that reported $450 million in revenues from the sale of computer programs in 1969, with a predicted rise to $2.5 billion by 1974.[25] Sales of programs produced by independent software firms had risen from $12 million to $320 million in the previous six years.[26] It appeared that, at least during the 1960s, the software industry did not need copyright to enjoy substantial growth and success.

B. Why the Economic Case for Software Copyright Was Uneasy in 1970

The fact that the software industry had grown so significantly in the 1960s without utilizing copyright was one reason *The Uneasy Case* was skeptical about the need for this protection for programs. But the article sought to explain in greater detail how this industry was able to

---

22 Breyer, *supra* note 1, at 344 (speculating about the low number of registrations). One factor that might explain the low registration numbers was that registration was not a precondition of copyright; it was thus possible that more programmers were interested in copyright than had registered their claims. *Id*. at 344 n.247. Moreover, courts had not yet clarified what scope of protection copyright might provide for programs, and the number of widely disseminated programs was still relatively small. *Id*.

Although Breyer may have been right to think that these factors contributed to the low number of registrations, there were at least two additional factors that likely deterred registration of programs. One was that the Office required programmers to deposit the full text of program source code in order to qualify for a registration certificate. *See* CIRCULAR 31D, *supra* note 18. This deposit of source code would have dissipated any trade secrets embedded in the programs, a form of legal protection on which software developers frequently relied. *See* Breyer, *supra* note 1, at 349 n.269 (noting the importance of trade secrecy for developers). A second factor was that the registration certificates expressed the Office's doubts about the copyrightability of programs. *See* CIRCULAR 31D, *supra* note 18, at 652, 652 n.72.

23 Breyer, *supra* note 1, at 344.

24 *Id.*

25 *Id*. at 344 n.246 (citing *EDP Industry Report*, EDP INDUSTRY REP. & MARKET REV. (Data Publ'g Co., Newtonville, Mass.), Mar. 12, 1970.

26 *Id*. (citing Patrick J. McGovern, *Free Competition and Illegal Restraints in Software*, *in* COMPUTERS-IN-LAW INST., GEORGE WASHINGTON UNIV., 1969 PROCEEDINGS: THE LAW OF SOFTWARE J-1, J-20 (1969)). Breyer further noted that IBM Corporation's ("IBM") decision to price software separately from hardware in the future would likely mean that the independent software business would grow significantly. *Id*.

function without copyrights.[27] It considered (1) whether developers could recoup the costs of producing software through other means besides the sales of copies, as the book industry did;[28] and (2) if copyists would even be able to sell unauthorized copies of software profitably.[29] It recognized that there was not one software industry, but rather several different sectors that were classified under the rubric of the software industry.[30]

One significant sector was systems software, which in 1970, was expected to account for approximately twenty-five percent of software-development costs.[31] Systems software was, "and should continue to be, created by hardware manufacturers and sold along with their hardware at a single price."[32] Systems software-development costs could therefore be recouped without the need for copyright protection. Such programs were typically capable of running on only that maker's hardware, which lessened the risk that a copyist could profitably sell unauthorized copies of that system software.[33]

Application programs, which accounted for about sixty percent of software development costs, tended to be "tailored to suit individual customer needs."[34] As a result, their development costs could be recouped through fees charged for customization.[35] Application program development was unlikely to be subject to unauthorized copying because "more than half of all computer time is accounted for by programs that computer users develop within their firms for their own use."[36] More generally usable application programs (i.e., ones that were not customized for particular end-user needs) tended to be sold in packages, which included documentation as well as installation and maintenance services.[37] Insofar as a computer user might be "buying services and expertise as much as he is buying a particular computer

---

27 *Id.* at 323–51.

28 *Id*. at 344–47.

29 *Id*. at 344–45.

30 *Id*. at 342–43.

31 *Id*. at 345.

32 *Id*. at 344. Although some firms, such as IBM, still bundle system software with hardware, not all do. *See infra* note 71 and accompanying text. Breyer did not foresee the possibility that independent software firms might be able to develop system software capable of running on multiple computers, such as Microsoft's Windows operating system which dominates today's computing ecosystem. *See infra* note 71.

33 Breyer, *supra* note 1, at 345.

34 *Id.*

35 *See id.*

36 *Id.*

37 *See id.*

program," a copyist was unlikely to attain a competitive advantage over the software's maker unless it could offer comparable services and expertise.[38] In any event, the software's maker was likely to have a lead-time advantage over copyists.[39]

Not only was the case for copyright uneasy because it appeared to be unnecessary for the software industry, but Breyer also warned that copyrighting programs would create some risks—and impose some costs—that should be considered before extending copyright protection to computer programs.[40] First, he worried that the scope of copyright in programs might prove to be either too weak or too strong.[41] If programmers could easily rewrite code, and thereby avoid infringement of the first developer's copyright, then little would be gained by the extension of copyright protection to programs.[42] And yet if courts were too quick to find infringement based on substantial similarities, there were risks that programmers would "avoid using even the algorithm of another's program [or] write new programs in unusual ways to lessen the risk of 'similarity.'"[43] This, according to Breyer, would be "wasteful."[44]

Breyer was also concerned about transaction-cost problems that might arise if one had to get permission for every borrowing from another's program, and standardization might be impeded if programmers felt compelled to adopt different ways of performing functions in order to avoid infringement.[45] Moreover, if copyright were used to protect program innovations, there would be a risk of excessive pricing of programs and anticompetitive conduct.[46] Finally, Breyer

---

38 *Id.*

39 *Id.*

40 *Id*. at 347–48; *cf*. CONTU REPORT, *supra* note 2, at 23–25 (dismissing concerns about economic risks of extending copyright protection to computer programs).

41 Breyer, *supra* note 1, at 347–48.

42 *Id*. Breyer used strong language in expressing this risk: if copyright was too thin, it "would risk emasculating the law." *Id*. This observation prefigured the Third Circuit's ruling in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*, 797 F.2d 1222, 1237 (3d Cir. 1986), in which the court expressed concern that thin copyright protection for programs would lead to underinvestment in software development. For a discussion of the reasons why cycles of under- and overprotection were likely to result from using copyright to protect innovations embodied in computer programs, see Samuelson et al., *supra* note 15, at 2356–61.

43 Breyer, *supra* note 1, at 348.

44 *Id*. Breyer thought that this risk could be mitigated by imposing a heavy burden of proof on those who claimed infringement based on substantial similarities. *Id.* at 348 n.264.

45 *Id*. at 347. Breyer quoted Professor Anthony Oettinger, a Harvard computer science colleague, about the potential for "disastrous consequences" if standardization was thwarted by innumerable variations in programs. *Id.*

46 "[T]he freedom from direct competition that copyright provides may allow some pro-

thought that the duration of copyright might be too long given that it "would apply to virtually all types of programs regardless of need" and the average commercial life of programs was likely to be considerably shorter than the copyright term.[47]

Breyer concluded that an empirical assessment of the software industry, considered as a whole, "cast considerable doubt on the present need for computer program copyright protection."[48] Moreover, for reasons explained in the next Section, Breyer did not find persuasive an argument in favor of a future need for program copyrights.

C. *Breyer's Predictions: The Uneasy Case for Software Beyond 1970?*

The Uneasy Case recognized that in coming years, there might be a substantial rise in the number of generally usable programs sold "off the shelf."[49] Yet, even then, Breyer was not convinced that copyright protection would be necessary to induce development of this software.[50] At the time, generally usable programs were being sold for ten to twenty percent of the price of developing them. This suggested that, even in the absence of copyright protection, "[i]t should therefore not prove difficult to organize a fairly small group of buyers to pay for the creation of a program they desire."[51]

Also undercutting the case for copyright protection for programs, in Breyer's view, was the predicted expansion of "time-sharing" software systems in coming decades.[52] This would give hundreds, or perhaps thousands, of computer users access to programs through a central computer system to which they would be connected by communications wires.[53] Moreover, computer hardware manufacturers

---

gram creators to charge prices considerably higher than needed to recover development expenses." *Id.* The fabulous wealth that software sales have generated for some entrepreneurs bears out this point. *See supra* notes 24–26 and accompanying text. But some would argue that it is the prospect of fabulous wealth that drives some entrepreneurs to produce software. *See, e.g.*, F. M. Scherer, *The Innovation Lottery*, *in* EXPANDING THE BOUNDARIES OF INTELLECTUAL PROPERTY: INNOVATION POLICY FOR THE KNOWLEDGE SOCIETY 3, 15–21 (Rochelle Cooper Dreyfuss et al. eds., 2001).

47 Breyer, *supra* note 1, at 348. Breyer believed that "the active life of the ordinary program [is] relatively short." *Id*. at 348 n.267.

48 *Id*. at 346.

49 *Id.*

50 *Id*. at 345–46. Breyer pointed to "pressures for greater program uniformity and compatibility" as factors that might contribute to the increase in sales of "off the shelf" software. *Id*. at 345.

51 *Id*. at 346.

52 *Id.*

53 *See id*. Time-sharing of computing systems was common in the 1970s, although this

would have incentives to develop and disseminate applications programs as promotional devices to induce people to purchase their machines.[54] Educators and government officials might also contribute to the development of generally usable programs, and they would not need copyright protection to induce them to do this.[55]

Although the case for a future need for program copyrights was far from compelling, Breyer maintained an open mind about the possibility that such a need might arise in the future.[56] *The Uneasy Case* articulated a set of five conditions that could make the case for program copyrights stronger: (1) a rise in the development of generally usable programs, (2) that were produced by independent software companies, (3) sold "off the shelf," (4) at low prices, (5) to large numbers of buyers.[57] Yet, even if those conditions came to pass, Breyer believed that "further empirical study [should be undertaken to] help to measure the real depth of that need [for copyright] by examining possible alternative methods of paying for such program development."[58]

D. Comparing Copyright to Other Forms of Legal Protection in The Uneasy Case

Although Breyer was skeptical about copyright as a form of legal protection for programs, he was not dead set against it. For example, he regarded copyright as preferable to patents for protecting program innovations.[59] Not only were patents of lesser value to programmers (because they are harder to get than copyrights and prior art searches

---

trend died out over time. *See infra* note 77 and accompanying text. One might, of course, consider the Internet today as a system for time-sharing of computing systems—one that is very much connected through wire—as well as wireless-communication technologies. But prescient as he was, Breyer did not invent, let alone foresee, the Internet. *Cf.* Breyer, *supra* note 1, at 346.

54 *See* Breyer, *supra* note 1, at 346. Breyer was perceptive in recognizing that the availability of applications programs could drive the sales of computer hardware. *Id.* The success of the IBM PC was, for instance, initially due largely to the availability of the spreadsheet program that served business accounting needs, namely, Lotus 1-2-3. *See infra* note 89 and accompanying text. Yet Breyer cannot be credited with the discovery of network effects as a significant factor in the success of software systems.

55 *See* Breyer, *supra* note 1, at 346. It is notable that Breyer was enough of an empiricist that he learned how to develop a flow chart and write an executable program. *Id*. at 341–42 & nn.235–36. One gets the sense that Breyer did not need copyright as an inducement to write that program; the prospect of tenure for writing a well-researched article was probably far more salient at the time.

56 *See id*. at 347.

57 *See id.*

58 *Id.*

59 *Id.*

would be very costly), Breyer also regarded them as more dangerous to the industry (because of their more potent exclusionary nature).[60]

Breyer also favored copyrights over trade secrecy as a way to protect programs.[61] Although trade secrecy protection was less potent than the exclusionary rights associated with patent protection, and would avert the transaction-cost problems Breyer foresaw with copyright,[62] Breyer questioned whether it would be an appropriate form of protection for mass-marketed software sold to large numbers of customers.[63] Trade secrecy also had the disadvantage of restricting dissemination of knowledge about program innovations.[64]

Finally, while *The Uneasy Case* did not mention a sui generis form of protection as a possible alternative for protecting programs, Breyer would likely have had an open mind to this idea, assuming empirical evidence and economic analysis supported it over copyright.

The Uneasy Case predicted that the software industry "may soon dwarf book publishing," and that if copyright was applied to software, it might need to be tailored to the special characteristics of this subject matter.[65] Although Breyer regarded the case for software copyrights as weak, he nevertheless believed that "details of a meaningful form of copyright protection [for software] could be worked out" if Congress chose to protect it that way.[66]

---

60 *Id*. at 348–49 n.268 ("[P]atent protection will provide IBM with the power to prevent competitors from using any of the ideas contained in their programs—a far more serious threat to competition than copyright's inhibition on copying them.") Given Breyer's concern, it is interesting to note that IBM was initially opposed to patent protection for program-related inventions. It joined an amicus curiae brief in opposition to the grant of a patent for an algorithm for transforming binary-coded decimals into pure binary form. *See* Gottschalk v. Benson, 409 U.S. 63, 63 (1972) (noting IBM's amicus brief urging reversal of the lower court ruling that Benson's algorithm was patentable). IBM may have been worried that if independent software developers patented software innovations, this might interfere with the ability of its customers to make optimal uses of IBM's computers. Over time, IBM changed its position on the patentability of software innovations. *See, e.g.*, Brief Amicus Curiae of International Business Machines Corporation in Support of Neither Party, Bilski v. Doll, 129 S. Ct. 2735 (2009) (No. 08-964), *available at* http://www.patentlyo.com/08-964-ibm.pdf.

61 Breyer, *supra* note 1, at 349–50 n.269.

62 *Id.*

63 The larger the number of people who know a secret, the less likely it can be claimed as a trade secret. *Id*.

64 Breyer noted that many software developers were in fact relying on trade secrecy for programs. *Id*. He regarded as "highly dubious" the argument that extending copyright protection to software would induce programmers to give up trade secrecy claims for programs. *Id*.

65 *Id*. at 350.

66 *Id*. at 343.

Forty years later, it is fair to say that these predictions have been borne out.[67] The software industry has indeed dwarfed the book publishing industry.[68] Copyright law did need to be tailored to deal with software protection, and details of meaningful protection of software were eventually worked out.[69] Yet, it is also fair to say that the state of software copyright law was in flux, and the scope of protection was uncertain and hotly contested—sometimes veering toward overly strong protection for programs, and sometimes toward underprotection, just as Breyer predicted—for the first fifteen years after Congress made copyright protection available for computer programs.[70]

II. THE ROLE OF COPYRIGHT IN THE PHENOMENAL GROWTH OF THE U.S. SOFTWARE INDUSTRY FROM 1970–2000

A. IBM's Unbundling Software from Hardware Systems as a Catalyst for the Software Industry's Growth

While Breyer was writing *The Uneasy Case*, a very important development for the U.S. software industry was taking place. In January 1970, under pressure from antitrust authorities, IBM Corporation ("IBM") decided to begin unbundling software from its hardware systems.[71] Previously, IBM had provided software free to customers of its hardware, a practice that other computer companies followed.[72] In *The Uneasy Case*, Breyer correctly anticipated that the unbundling decision would lead to a substantial growth in the independent software-developer market.[73] Before this, the independently developed software products that succeeded tended to be "those that satisfied needs not yet anticipated by the computer manufacturers."[74]

The IBM unbundling decision was a turning point for the software industry because it allowed independent developers to enter the market with products that competed with or complemented IBM's

---

67 *See infra* Part II.

68 *See infra* notes 78–87 and accompanying text.

69 *See infra* Part II.

70 *See* infra Part II.

71 MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 6 (2003). In the late 1960s, IBM had a seventy-percent share of the U.S. computer market. *Id*. at 109. Antitrust authorities charged that "IBM was competing unfairly with other manufacturers by supplying software and other services, without regard to their true cost, in order to win an account. In effect, IBM was selling below cost—unquestionably an antitrust violation." *Id*.

72 *See id*. at 6.

73 Breyer, *supra* note 1, at 344 n.246.

74 CAMPBELL-KELLY, *supra* note 71, at 6.

hardware and software products.[75] It also became possible to provide various kinds of services to IBM customers, including "product customization, user training, and regular upgrades," which proved to be "unexpected sources of income for which the pioneers of the industry had not initially planned."[76] While computing support services became a lucrative business, time-sharing of software, contrary to Breyer's prediction, faded away.[77]

B. The Numbers: Growth in the Software Industry: 1970–2000

In the decade that followed IBM's unbundling decision, the independent software industry grew substantially. In 1970, there were approximately 1400 firms that produced software or provided computer services. By 1975, the number of such firms had almost doubled, and by 1980, there were more than three times as many software firms as in 1970.[78] In those same years, the total revenues in the software industry rose from $1.9 billion to $14.9 billion.[79] Most of these revenues, however, were derived from computer *services*. In 1970, for instance, the computer service industry amassed $1.46 billion, whereas software sales drew in only about $440 million.[80] By 1980, revenues from sales of software products had risen to $6.1 billion, but this was still considerably less than the $8.8 billion in service revenues.[81]

It was not until the early 1980s that the U.S. software industry really took off.[82] By 1985, industry revenues had risen to $23.6 billion; and by 1990, revenues reached an unprecedented high of $51.3 billion.[83] While computer services were still a substantial part of the market in 1990,[84] software product revenues exceeded computer-service revenues that year.[85] Rapid growth continued through the

---

75 *See id.*

76 *Id.* at 6–7.

77 *Id.* at 124. The principal reason for the demise of time-sharing was the rise in the market for non-mainframe computers, including PCs. *Id.* at 122–23.

78 *Id.* at 18–19 tbl.1.2.

79 *Id.*

80 *Id.*

81 *Id.*

82 *Id.*

83 *Id.*

84 In 1985 and 1990, revenues from computer services were $16.3 billion and $25.1 billion, respectively. *Id.*

85 *Id.*

1990s.[86] By 2000, user expenditures on software products topped $100 billion, and programming-service revenues exceeded $33 billion.[87]

Much of the more than eight-fold growth in the software industry between 1980 and 1990 was due to the enormous popularity of personal computers and "killer apps" that could run on these computers.[88] Lotus Development Corporation, for instance, went from receiving $53 million in revenues in 1983, its first year in business, to $1.15 billion by 1995.[89] Intuit experienced a similar trajectory; its revenues rose from $18.6 million in 1989, the year it introduced Quicken to the market, to $1.09 billion by 2000.[90] Oracle generated $13 million in revenues from sales of its database software in 1984; only sixteen years later, its revenues went above $10 billion.[91] Yet, it was Microsoft Corporation ("Microsoft") that enjoyed the most substantial success of all. Between 1983 and 1995, Microsoft's software revenues grew from $70 million to $7.27 billion.[92]

C. Was Copyright Protection Responsible for this Growth?

The availability of copyright as a form of legal protection for computer programs does not deserve all of the credit for the phenomenal growth of the software industry,[93] but it did play a nontrivial role in the industry's success. Once early court rulings established that copyright could be used to protect machine-executable forms of programs from exact copying, companies in the software industry knew that they could rely on copyright for the protection of this important aspect of their products.[94] True to Breyer's predictions,[95] software firms largely chose to distribute their products only in machine-executable form, leaving most of the commercially valuable know-how embedded in programs to be protected as trade secrets.[96]

---

86 *See id*. at 15 tbl.1.5.

87 *Id*. at 15 tbl.1.1.

88 *See id*. at 202–28.

89 *Id*. at 252 tbl.8.8.

90 *Id*. at 296 tbl.9.6.

91 *Id*. at 186 tbl.6.7.

92 *Id*. at 235 tbl.8.2.

93 *Id.* at 303–11 (providing an overview of factors that contributed to the success of the software industry).

94 *See, e.g*., Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249–55 (3d Cir. 1983) (upholding the validity of Apple's copyrights in its operating-systems programs).

95 *See supra* note 22 and accompanying text.

96 *See, e.g*., CAMPBELL-KELLY, *supra* note 71, at 107–08. Patents played a relatively modest role in the legal protection of computer program innovations between 1970 and the mid-1990s. *See infra* note 220 and accompanying text.

In *The Uneasy Case*, Breyer drew an important distinction between systems software and applications software in terms of strategies that software developers could use to recoup their research and development ("R&D") investments.[97] Between 1970 and 2000, user expenditures on systems software increased from $150 million per year to $41.7 billion per year.[98] In 1970, users spent sixty percent of software expenditures on systems software; in 1980 and 1990, users spent about as much on systems software as on application programs; yet by 2000, expenditures on systems software had dropped to under forty percent.[99]

The applications market, by contrast, grew substantially after 1970. Expenditures for software-application products in 1970 were roughly $100 million.[100] Only a decade later, these expenditures had risen to $1.4 billion; two decades later, to $17.7 billion; and three decades later, to $63 billion.[101] The applications market included business productivity software, such as spreadsheets, word processing, and database-management software, as well as home and recreational software, such as videogames. Many specialized applications markets also developed for particular industry segments, such as stock trading software. Successful application developers tended to enjoy substantial profits, for once there have been enough sales to recoup the initial R&D costs, additional sales were almost entirely profits, given that the cost of replicating software products is very low.[102]

Although the distinction between systems and applications software continued to be salient in the decades after the publication of *The Uneasy Case*, the systems software sector evolved in a somewhat different manner than Breyer foresaw. The following two Sections evaluate the role of copyright in the markets for systems software and applications software, respectively.

1. Copyright and the Market for Systems Software

In 1970, each computer-hardware firm tailor-made systems software for its machines; under these conditions, Breyer believed that the costs of developing systems software could be recouped from the

97 Breyer, *supra* note 1, at 342–45.

98 CAMPBELL-KELLY, *supra* note 71, at 14–15.

99 *Id.*

100 *Id.*

101 *Id.*

102 *Id.* at 124.

sale of the hardware with which the systems software was bundled. Hence, copyright protection was unnecessary.

However, four noteworthy developments altered the evolution of the market for systems software such that copyright protection was, in fact, beneficial. First, in the 1970s and 1980s, the success of certain computer hardware platforms—the Apple II computer, for instance, and the IBM 360 series computers—induced some competitors to "clone" systems software for popular machines in order to take advantage of the software that independent firms were developing for those platforms.[103] The successful entry of clone systems software posed a risk of undermining the usual recoupment strategy for such software. Second, some firms that did not manufacture hardware—most notably Microsoft—were successful in entering the systems software market.[104] Third, the introduction of graphical user interfaces ("GUIs") in the 1980s made systems software more visible to users than in earlier periods.[105] Fourth, the distinction between systems software and applications blurred further as some makers of systems software began to integrate functions that previously had been performed by applications programs into the core of their systems software.[106]

Cloning attracted copyright lawsuits. Franklin Computer, which cloned the hardware and software of the Apple II platform, was unsuccessful in defending against Apple's copyright lawsuits when it argued, among other things, that it was necessary to copy Apple operating systems programs exactly in order to provide the customers of their competing computers with machines compatible with programs written for the Apple II platform.[107] Compatibility was, in the court's view, "a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged," and hence, the cloner's idea/expression merger defense to infringement failed.[108] Had this defense succeeded, it might have affected Apple's ability to recoup its R&D investments in systems software.

---

103 Cloning was generally aimed at reimplementing the functionality of the software being cloned. Samuelson et al., *supra* note 15, at 2395–98.

104 *Id.* at 2372.

105 *Id*. at 2377, 2410 n.407.

106 *See infra* note 131 and accompanying text.

107 *See, e.g*., Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1251-52 (3d Cir. 1983).

108 *Id.* at 1253.

Yet, even when competitors implemented system software functionality in different program code, as Fujitsu did in competing with IBM so that programs written for IBM machines could also run on Fujitsu's machines,[109] copyright challenges ensued. IBM and Fujitsu eventually settled their legal dispute through a complex settlement agreement under which IBM granted Fujitsu the right to continue to develop IBM-compatible systems software in return for paying license fees to IBM.[110] These license fees helped IBM recoup its investment in systems software that might have been undercut by a successful unlicensed clone of its systems software. Ambiguity about the scope of copyright protection for systems software at the time of this dispute[111] (i.e., did copyright protect IBM interfaces and other internal design elements of its systems software, or only the exact code of the programs?) contributed to the motivation of both IBM and Fujitsu to settle this dispute, for a great deal was at stake for both sides in how this issue was resolved.[112]

More surprising, however, was the rise of a systems software powerhouse, Microsoft, whose operating system ("OS") software became—and still is—a de facto industry standard, despite the fact that Microsoft has never manufactured computers.[113] In this respect, Breyer's theory that the costs of developing systems software could be recouped through the sale of computers running that software[114] did not apply. Microsoft attained its privileged status as a result of a decision by IBM not to develop its own OS when it introduced the IBM PC into the market. IBM instead contracted with Microsoft to use the latter's MS-DOS program as the PC's systems software.[115] Microsoft tailored the design of its OS to the architecture of Intel semiconductors. Because any other computer company could buy Intel chips as the microprocessors for their machines and then license MS-DOS as systems software, it was possible to manufacture IBM-PC compatible computers to compete with IBM's PC.[116] Microsoft's strategy for re-

109 *See, e.g.*, ANTHONY L. CLAPES, SOFTWARE, COPYRIGHT, AND COMPETITION: THE "LOOK AND FEEL" OF THE LAW 166–72 (1989) (discussing the IBM-Fujitsu dispute and its resolution).

110 *Id.*

111 *See* Alisa E. Anderson, *The Future of Software Copyright Protection: Arbitration v. Litigation*, 12 HASTINGS COMM. & ENT L.J. 1, 17–18 (1989).

112 *See id.* at 23.

113 *See, e.g.*, CAMPBELL-KELLY, *supra* note 71, ch. 8 (discussing Microsoft's history as a software developer).

114 Breyer, *supra* note 1, at 346.

115 *See, e.g.*, CAMPBELL-KELLY, *supra* note 71, at 240.

116 *Id.* at 231–51.

couping its R&D investment in MS-DOS was quite different than what Breyer envisioned for the systems-software market, and copyright was important to that strategy.

The market for IBM PCs and IBM-compatible computers grew rapidly,[117] in part because independent software companies were drawn to developing programs for this platform because of the large installed base. This in turn made the platform more desirable for consumers, further enlarging the installed base.[118] Although some independent software developers tried to compete with Microsoft's OS,[119] these efforts were not successful, owing in part to aggressive (and, arguably, anticompetitive) tactics used by Microsoft to induce computer manufacturers to install Microsoft's OS on their machines.[120] Between 1986 and 1995, Microsoft's revenues from the sale of systems software grew from $100 million to close to $2 billion a year.[121]

In the late 1980s, Microsoft introduced the Windows OS.[122] Unlike MS-DOS, which had featured a user-unfriendly command-line interface, Windows adopted a GUI through which users could more easily interact with their computers and applications software.[123] Although Microsoft initially got a license from Apple to develop a GUI

---

117 In 1984, IBM received $4 billion in revenues from sales of PCs. JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL: INTELLECTUAL PROPERTY AND INTEROPERABILITY IN THE GLOBAL SOFTWARE INDUSTRY 30 (1995).

118 This kind of positive feedback effect came to be known as "network effects." *See* CAMPBELL-KELLY, *supra* note 71, at 264–65. Platforms generally became more desirable to consumers as the number of applications available for that platform rose. *Id.*

119 At one time, there were two dozen operating systems competing with MS-DOS, some of which were technically superior to MS-DOS. *Id*. at 241.

120 In 1994, the Antitrust Division of the U.S. Department of Justice charged Microsoft with violating the antitrust laws based on its restrictive OS licensing practices of original equipment manufacturers ("OEMs"). *See* United States v. Microsoft Corp., 980 F. Supp. 537, 539 (D.D.C. 1997), *overruled on other grounds*, 147 F.3d 935 (D.C. Cir. 1998). OEMs had been required to pay Microsoft a royalty for each IBM-compatible computer they sold if they wanted to qualify for a discount, regardless of whether they installed the Microsoft OS or another firm's systems software on each machine. *Id.* This litigation ended with a consent decree aimed at overcoming barriers to entry arising from Microsoft's OEM licensing practices. *Id.*; *see also* BAND & KATOH, *supra* note 117, at 35–36.

121 CAMPBELL-KELLY, *supra* note 71, at 233, 242.

122 *Id.* at 249–50.

123 *See* Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1019–20 (N.D. Cal. 1992), *aff'd*, 35 F.3d 1435 (9th Cir. 1994). Command-line interfaces, such as MS-DOS, required users to remember and enter specific names to invoke particular functions that users wanted to be performed. *See id.* at 1017–20. GUIs provided visual representations such as icons to represent functions (e.g., trash can icons to represent deletion functions) and windowing capabilities so that more than one applications program could be displayed on computer screens at a time. *Id.* Prior to the introduction of GUIs, systems software was, for the most part, invisible to users. *Id.*

for Windows 1.0 that was similar to Apple's Macintosh GUI, Microsoft did not update that license for later versions of Windows or to include later-adopted features of the Macintosh interface.[124] Because the main visual elements of Windows 2.0 were very similar to the Macintosh GUI, Apple sued Microsoft for copyright infringement, claiming that it had stolen the "look and feel" of the Apple interface.[125] In the course of the litigation with Microsoft, Apple emphasized the aesthetic design of its GUI and relied on copyright cases extending protection to the "look and feel" of graphical works.[126] Microsoft compared its and Apple's GUIs to automobile dashboards, which, because of their functionality, are ineligible for copyright protection.[127] The courts ultimately ruled in favor of Microsoft, finding the dashboard metaphor more convincing than the aesthetic-design metaphors that Apple had proffered.[128]

These decisions contributed to the view that a second comer could reimplement the functionality of a popular program in different code without running afoul of copyright law, and even strong visual similarities in the "look and feel" of two programs might be excusable if they performed many of the same functions.[129]

A key difference between the systems software market in 1970 and the market in the 1990s was the self-conscious integration of application program functionality into systems software as a means to compete with application developers.[130] The most significant example is Microsoft's decision to integrate its Internet Explorer browser into the Windows OS.[131] One motivation to undertake this integration was to ward off the competitive threat of a then-widely adopted browser,

---

124 *Id.* at 1015; *see also* CAMPBELL-KELLY, *supra* note 71, at 250.

125 CAMPBELL-KELLY, *supra* note 71, at 250.

126 *Apple Computer, Inc.*, 799 F. Supp. at 1019–20; *see also, e.g.*, Roth Greeting Cards v. United Card Co., 429 F.2d 1106, 1110 (9th Cir. 1970) (finding that the copyright for greeting cards was infringed because of similarities in the "look and feel" of the plaintiff's and defendant's cards).

127 *Apple Computer, Inc.*, 799 F. Supp. at 1016, 1023.

128 *Id.* at 1026–27, 1047; *see also* Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1445 (9th Cir. 1994).

129 The Ninth Circuit rigorously filtered out unprotected elements in assessing copyright infringement for computer-program user interfaces. *Apple Computer, Inc.*, 35 F.3d at 1446–47. Insofar as program interfaces were composed primarily of unprotected elements, the Ninth Circuit opined that the defendant's work would need to be "virtually identical" to the plaintiff's before a court would find infringement. *Id*. at 1442.

130 *See, e.g.*, KEN AULETTA, WORLD WAR 3.0: MICROSOFT AND ITS ENEMIES, at xix–xxi (2001).

131 *Id.*

Netscape's Navigator.[132] Netscape's browser had not only attained a significant competitive position vis-à-vis Microsoft's Internet Explorer in the mid-1990s, but it also seemed to be (or, seemed capable of) evolving into a platform for networked computing that might have made it possible to bypass Microsoft's OS.[133]

In *The Uneasy Case*, Breyer warned that copyrights for software might be exercised in anticompetitive ways,[134] though he did not anticipate (nor did anyone else) that firms such as Microsoft might claim—as it did in *United States v. Microsoft Corp.*[135]—that the copyrights in their software entitled them to exercise unfettered control over their products without running afoul of antitrust laws.[136] In *Microsoft*, the District Court for the District of Columbia ruled that Microsoft violated the antitrust laws when it bundled its browser software into the Windows OS for the purpose of maintaining its monopoly in the OS market and preventing original equipment manufacturers and users from separating them.[137] The court also rejected Microsoft's copyright-based defense.[138] Yet, even after extensive negotiations between Microsoft and antitrust officials to adopt a meaningful remedy for this violation,[139] the firm's de facto monopoly in the OS market has persisted.[140]

2. *Copyright and the Market for Applications Software*

In *The Uneasy Case*, Breyer expressed skepticism about the need for copyright protection for application programs because, in 1970, most applications were custom developed for particular clients.[141] Owners of computers often developed application programs to meet their needs, and time-sharing of computer resources was a foreseeable

---

132 *Id.*

133 *Id.* at 55–56.

134 *See supra* note 46 and accompanying text.

135 United States v. Microsoft Corp., 87 F. Supp. 2d 30 (D.D.C. 2000), *aff'd in relevant part*, 253 F.3d 34 (D.C. Cir. 2001) (per curiam).

136 *See Id.* at 40.

137 *Id.* at 38–40, 47.

138 *Id*. at 62–64.

139 AULETTA, *supra* note 130, at 340–68. Part III shows that Microsoft's dominance in the software industry has been mitigated by developments in networking software systems.

140 *But see* Jon Brodkin, *Windows on Verge of Dropping Below 90% Market Share*, NETWORK WORLD (Jan. 13, 2011, 2:25 PM), http://www.networkworld.com/news/2011/011311-windows-on-verge-of-dropping.html ("Microsoft's continued dominance of the desktop operating system market will likely not be enough to keep Windows' total share above 90%, because the proliferation of smartphones and tablets is changing the definition of what a personal computer is.").

141 *See supra* notes 34–39 and accompanying text.

trend.[142] At that time, general-purpose application programs were a very small part of the software market, owing in part to the long-standing practice by computer manufacturers of bundling software and hardware.[143] Yet, Breyer recognized that certain changes would strengthen the case for copyrighting application programs; these changes did occur in the decades after *The Uneasy Case* was published. Specifically, there was a rise in the development of generally usable programs produced by independent software companies sold "off the shelf" at low prices to large numbers of buyers.[144]

The success of certain applications programs in the early 1980s to the mid-1990s made competitive copying easy to accomplish, and this in turn led to considerable litigation about applications program copyrights. The least controversial precedents were those that established that exact copying of videogame program code and of audiovisual aspects of videogames infringed copyrights.[145] Two issues provoked the greatest controversy. First whether copyright protection for application programs extended to the "structure, sequence, and organization" ("SSO") of these programs—such as internal interfaces, algorithms, and data structures—or to the "look and feel" of programs.[146] Second whether copying another firm's code in the course of reverse engineering that code to learn how to make an application program that would successfully interoperate with that firm's other software was copyright infringement or fair use.[147] The cases dealing with these issues addressed the risk raised in *The Uneasy Case for Copyright* that copyright would either over- or under-protect programs. However, as Breyer predicted, the basic contours of copyright scope for programs evolved over time and are now quite stable.

a. The SSO and "Look and Feel" Controversies

The SSO question was most directly addressed in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[148] *Whelan* involved a dispute between two former partners who had worked together to develop a program to manage various aspects of dental laboratory

142 *See supra* notes 52–53 and accompanying text.

143 *See* Breyer, *supra* note 1, at 344 n.248.

144 *Id.* at 347.

145 *See, e.g.*, Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870, 877 (3d Cir. 1982) (finding infringement of videogame copyrights).

146 *See, e.g.*, Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1231 (3d Cir. 1986).

147 *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514–15 (9th Cir. 1992).

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

businesses.[149] After a feud, Jaslow decided to develop his own version of the Dentalab program for the IBM PC, which he believed would make the program more marketable than the computer for which Whelan had tailored the program.[150] Although each partner wrote in different programming language and used different algorithms, the overall structure of the programs was similar, as were some data and file structures.[151] Further, the two programs performed some of the same functions in the same manner.[152] The district court held that Jaslow infringed Whelan's copyright because the two programs had similar overall structures and functioned very similarly (i.e., they had a similar "look and feel").[153] Jaslow's principal defense on appeal was that copyright protection only extended to source and object code, and not to program SSO or "look and feel."[154] The Third Circuit affirmed the finding of infringement in an opinion that vindicated Whelan's claims.[155]

The Third Circuit's *Whelan* decision characterized computer programs as "literary works" and reasoned that because copyright law had long protected nonliteral elements (i.e., structure and organization) of other types of literary works, such as novels and plays, it should protect the SSO of programs as well.[156] The court also opined that program SSO should be protectable by copyright law as long as there is more than one way to structure a program to achieve the program's functions.[157] The court stated that the general purpose or function of a program could not be protected by copyright law because it was an uncopyrightable idea, and if there was only one way to structure a program to perform particular functions, then the "idea" of that function and its structural "expression" should be considered "merged."[158] Once merged, this structure could no longer be protected by copyright law.[159] Finally, the court endorsed copyright protection for the "look and feel" of programs, which seemingly included the manner in which the programs behaved (i.e., the sequence of

149 *Id.* at 1225–26.

150 *Id.* at 1226.

151 *Id*. at 1228.

152 *Id.*

153 *Id*. at 1228–29.

154 *Id*. at 1233.

155 *Id*. at 1248.

156 *Id*. at 1234.

157 *Id*. at 1236.

158 *Id.*

159 *Id.*

screen displays during program operations and how programs performed their functions).[160]

The court was persuaded that unless broad copyright protection was available for program SSO and the "look and feel" of a program—both of which were costly and difficult to develop yet cheap to copy—there would be not enough legal protection to provide proper incentives to develop computer programs.[161] The Third Circuit thus invoked the risk of underprotection, a risk that Breyer identified in 1970, as a justification for broader protection. In doing so, however, the court instead fulfilled Breyer's fear of overprotection.[162]

Although courts cited *Whelan* approvingly in a number of subsequent software copyright cases,[163] its test for software copyright infringement soon came under considerable criticism.[164] The most important decision to abjure *Whelan*'s approach to software copyright infringement was the Second Circuit's decision in *Computer Associates International, Inc. v. Altai, Inc.*[165] Computer Associates ("CA") and Altai were competitors in the market for scheduling software that was designed to run on two types of IBM machines.[166] Altai hired a former CA employee, Claude Arney, to work on developing a new version of its scheduling program Zeke.[167] Arney proposed that the best way to redesign Zeke to make it compatible with two different

160 *Id*. at 1228, 1247. Program behavior is an important part of the value of programs, but because of its largely functional character, copyright protection should not be available to it. *See* Samuelson et al., *supra* note 15, at 2320–41, 2347–56.

161 *Whelan*, 797 F.2d at 1237.

162 *See supra* notes 40–44 and accompanying text. In 1984, I warned that the functionality of computer programs would make it difficult to apply copyright law to programs and criticized the analogy of programs to literary works such as novels. *See* Samuelson, *CONTU Revisited*, *supra* note 2, at 727–53; *see also* J. H. Reichman, *Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research*, 42 VAND. L. REV. 639, 690–96 (1989) (noting that traditional copyright defenses might be ineffective in the context of computer software).

163 *See, e.g.*, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 52, 55, 68 (D. Mass. 1990) (relying on *Whelan* as support for finding a spreadsheet programs' user-interface design copyrightable); Pearl Sys., Inc. v. Competition Elecs., Inc., 8 U.S.P.Q.2d (BNA) 1520, 1524 (S.D. Fla. 1988) (relying on *Whelan* to find program user interface copyrightable).

164 *See, e.g.*, Donald S. Chisum et al., *Last Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 20 (1989); David Nimmer et al., *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625, 629–30 (1988); Pamela Samuelson, *Reflections on the State of American Software Copyright Law and the Perils of Teaching It*, 13 COLUM.-VLA J.L. & ARTS 61, 63 (1988).

165 Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

166 *Id.* at 698–701.

167 *Id.* at 699.

IBM operating systems was to build a new compatibility component for Zeke.[168] That is, Arney planned to build a subprogram (ultimately named Oscar), that would transpose Zeke's commands for specific tasks into the appropriate format so that each operating system could, in turn, properly instruct the IBM hardware to carry out Zeke's scheduling program commands.[169] This new design would avoid the need to customize each module of Zeke for the two operating systems; if Altai wanted to adapt Zeke in the future to be compatible with additional operating systems, the developer would only need to rewrite parts of Oscar, not the whole of Zeke.[170] Unbeknownst to Altai, CA had adopted the very same approach in the latest version of its CA-Scheduler program, a project on which Arney had worked when he had been in CA's employ.[171]

Altai produced and shipped the Oscar-enhanced version of Zeke until CA sued Altai for copyright infringement and trade secret misappropriation, alleging that Oscar contained code misappropriated from CA-Scheduler.[172] After being notified of the lawsuit, Altai executives asked Arney about the charges, and Arney confessed that he had taken a copy of his former employer's source code when he left the firm and had directly copied portions of this code while developing Oscar.[173]

Altai took immediate steps to purge Oscar of the tainted code.[174] First, Altai discerned which parts of the Oscar code had been directly copied from CA-Scheduler. Second, the company assigned a fresh engineering team to document the interfaces that Oscar needed to implement to interoperate successfully with the IBM systems. Third, a new team of Altai programmers revised Oscar with new code that implemented these interfaces.[175] Altai then began selling the revised

---

168 *Id.* at 699–700.

169 *Id*. Interfaces are important elements of computer programs because they are specially designed to enable the exchange of information between programs or program components. Like CA-Scheduler, Zeke had to interoperate with the systems software of two IBM mainframe computers, the interfaces for which were not the same. The Oscar subprogram, in essence, translated Zeke's commands into a format that each IBM OS could comprehend. The IBM hardware could then carry out the scheduling program's commands correctly. *See generally* Michael A. Jacobs, *Copyright and Compatibility*, 30 JURIMETRICS J. 91 (1989) (discussing program interfaces and the commercial importance of compatibility).

170 *Altai*, 982 F.2d at 699.

171 *Id.* at 699–700.

172 *Id.* at 700.

173 *Id.*

174 *Id.*

175 *Id.*

Zeke to new customers and offering it as a free upgrade to its existing customers.[176]

Altai accepted liability for the code directly copied from CA-Scheduler, but believed that the rewrite of Oscar had immunized it from further copyright liability.[177] CA, however, asserted that the revised Oscar program was still substantially similar in SSO to the compatibility subprogram of CA-Scheduler, particularly in the manner in which Altai structured the program interfaces.[178] CA relied heavily on *Whelan* and its progeny in arguing that the revised Oscar program infringed its copyright in CA-Scheduler.[179] It pointed to substantial similarities between the compatibility components of Zeke and CA-Scheduler, especially as to their parameter lists (i.e., lists of information that needed to be sent and received by subroutines of the affected programs).[180] These elements of program SSO had been designed carefully and precisely, which made them costly to develop and commercially significant.[181] CA argued that incentives to invest in software development would be undermined if competitors such as Altai could appropriate program SSO without fear of liability.[182] Moreover, parameter lists and other SSO elements of program interfaces are complex and detailed, not abstract in content. Therefore, they seemed to be protectable under *Whelan*.[183] CA argued that, in view of the SSO similarities, the revised Oscar still infringed the copyright in CA-Scheduler.[184]

Despite CA's reliance on *Whelan* and incentive-based arguments—arguments that had succeeded in the past—Altai convinced the trial and appellate courts to conceptualize computer programs as utilitarian works that were meaningfully different from novels and plays.[185] At best, utilitarian works enjoy "thin" copyright protection—

---

176 *Id.*

177 *Id*. at 701.

178 *Id.* at 702–03. Interfaces of computer programs are unquestionably parts of program SSO. *Id.* The *Altai* case, however, posed the question of whether similarities in program interfaces could be the basis for claims of copyright infringement.

179 Reply Brief for Plaintiff-Appellant at iv, *Altai*, 982 F.2d 693 (No. 91-7893).

180 *Id.* at 6–7; *see also Altai*, 982 F.2d at 697–98.

181 *Altai*, 982 F.2d at 698.

182 *Id.* at 711–12.

183 *Id.* at 713–15.

184 *Id.*

185 *Id.* at 704 ("The essentially utilitarian nature of a computer program further complicates the task of distilling its idea from its expression.").

that is, protection against only exact or near-exact copying.[186] The court recognized that the design choices of programmers are often constrained by the mechanical specifications of the computer hardware on which a program was designed to run and by systems software interfaces, such as the IBM protocols that enabled programs like CA-Scheduler and Zeke to exchange information and interoperate with the IBM software.[187] It was necessary for the parameter lists of CA-Scheduler and Zeke to be substantially similar because they both provided scheduling services to their customers and interoperated with the IBM programs.[188]

The *Altai* decision rejected CA's claims of copyright protection in interfaces and adopted a now widely used three-step test for assessing claims of copyright infringement in computer programs.[189] The first step involves constructing a hierarchy of abstractions, from most abstract to most detailed, for the plaintiff's program.[190] The second step involves filtering out from the analysis various elements of the program that are beyond the scope of copyright protection.[191] The third step involves comparing any remaining "golden nuggets" of expression in the plaintiff's program with the defendant's program to determine if the defendant copied substantial amounts of expression from the plaintiff's program.[192]

Application of this abstraction-filtration-comparison test generally results in programs having thin copyright protection. The second step narrows the scope of protection by filtering out three kinds of unprotected elements: (1) elements of program design dictated by efficiency;[193] (2) design choices constrained by external factors, such as the hardware and software with which the program was designed to operate, demands of the industry being served, and widely accepted programming practices;[194] and (3) public domain elements of programs, such as commonplace programming techniques, ideas, and

---

186 *Id*. at 704–05 (comparing computer programs to useful arts such as recipes and bookkeeping systems).

187 *Id*. at 709–10, 715.

188 *Id.*; *see also* Brief of Defendant-Appellee at 10–11, *Altai*, 982 F.2d 693 (No. 91-7893).

189 *Altai*, 982 F.2d at 706.

190 *Id.* at 707.

191 *Id.*

192 *Id.* at 710.

193 *Id*. at 707–09. As a hypothetical matter, there may be many ways to achieve certain program functions, but efficiency considerations will often narrow the range of practical solutions. *Id.* at 708. Because programmers are constantly striving to achieve efficiency, adopting the same efficient solution may be the product of independent work, not of copying.

194 *Id*. at 709–10.

know-how.[195] Since the *Altai* decision, the scope of copyright protection has become even thinner, as courts now also filter out functional design elements such as procedures, processes, systems, and methods of operation.[196]

In *Altai*, the court reasoned that the test it set forth "not only comports with, but advances the constitutional policies underlying the Copyright Act."[197] But, even if CA was right that thin copyright protection for programs might undermine incentives to invest in program development, its argument was inconsistent with a recent Supreme Court ruling that rejected similar incentive-based arguments for *broad* copyright protection of factual works.[198] The Second Circuit asserted that to extend broad copyright protection to program SSO would "have a corrosive effect on certain fundamental tenets of copyright doctrine."[199] Because copyright seemed ill suited to protect program innovations, the court in *Altai* suggested that Congress consider whether programs should have additional intellectual property protection.[200] It also suggested that patents might be a more suitable form of protection for program SSO.[201]

After *Altai*, courts became more openly skeptical about claims of copyright protection for the "look and feel" of programs, as such claims typically sought to protect the now unprotected utilitarian aspects of programs.[202]

The *Altai* decision responded to the overprotection risk that *Whelan* posed and provided a more nuanced and technically sophisticated framework for assessing software copyright infringement claims. This decision was an important step in working out the contours of software copyright law, as Breyer predicted.[203]

---

195 *Id*. at 710.

196 *See, e.g*., Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 836 (10th Cir. 1993).

197 *Altai*, 982 F.2d at 711.

198 *See* Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 359 (1991).

199 *Altai*, 982 F.2d at 712. The court criticized *Whelan* for its unduly broad conception of the scope of copyright in computer programs, for its reliance on metaphysical distinctions rather than practical considerations, and for its outdated comprehension of computer science. *Id*. at 705–06.

200 *Id*. at 712.

201 *Id.*

202 *See, e.g*., Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995) (rejecting claim of copyright infringement based on "look and feel" of spreadsheet program); Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435, 1439 (9th Cir. 1994); *see also supra* notes 107–08 and accompanying text.

203 *See* Breyer, *supra* note 1, at 345–46.

b. The Reverse-Engineering Controversy

Closely related to the SSO-in-interfaces controversy resolved in *Altai* was whether making copies of program code for the purpose of gaining access to information necessary to achieve interoperability was fair use.[204] A legal dispute addressing this question arose after Accolade, Inc. ("Accolade") developed an unlicensed videogame program to run on the Sega Genesis platform.[205] Accolade had approached Sega Enterprises, Ltd. ("Sega") about a licensing deal, but the parties were unable to reach a satisfactory agreement. Accolade thereafter reverse engineered Sega programs to discern how its interfaces were structured.[206] Accolade then used the information obtained from reverse engineering to write code so that its videogame would run on the Genesis console.[207] Sega sued Accolade for infringement, arguing that copies of Sega's code made in the course of reverse engineering were unlawful reproductions of its copyrighted works.[208]

In *Sega Enterprises Ltd. v. Accolade, Inc.*,[209] the Ninth Circuit embraced *Altai*'s conceptualization of computer programs as utilitarian works that were eligible for only thin copyright protection.[210] It also endorsed *Altai*'s ruling that program interfaces were elements of programs that copyright law did not protect.[211] The court ruled that reverse engineering of program code for a legitimate purpose, such as extracting interface information to make a compatible program, did not infringe any copyright in that code.[212] The court reasoned that:

> If disassembly of copyrighted object code is *per se* an unfair use, the owner of the copyright gains a *de facto* monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection by Congress. In order to enjoy a lawful monopoly over the idea or functional prin-

---

204 *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525–26 (9th Cir. 1992) (addressing the reverse-engineering issue two months after the *Altai* decision).

205 *Id.* at 1514.

206 *Id.* at 1514–15.

207 *Id.*

208 *Id.* at 1515.

209 Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

210 *Id.* at 1527 ("Under the Copyright Act, if a work is largely functional, it receives only weak protection."); *see also* Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 709–10, 715 (2d Cir. 1992).

211 *Altai*, 982 F.2d at 704–05; *see also Sega*, 977 F.2d at 1524–26 (discussing functional requirements for achieving compatibility with other programs).

212 *Sega*, 977 F.2d at 1527–28 (holding that reverse-engineering copies qualified as fair use).

> ciple underlying a work, the creator of the work must satisfy the more stringent standards imposed by the patent laws.[213]

The court also held that even copying of exact code from another's program would not constitute infringement insofar as that code was essential to achieving interoperability.[214]

Sega implied that the only reliable means for protecting functional designs, such as interfaces, was by patenting them.[215] Because *Sega* allowed unlicensed developers to reverse engineer other firms' code to extract interface information—and other elements of programs that were beyond copyright scope[216]—the decision seemingly threatened first developers' efforts to protect their programs' internal design elements as trade secrets.[217] Unlike copyrights, patents can be used to protect program interfaces, as patent law does not have a "merger" doctrine.[218] Hence, if there is only one way to achieve a particular function, and a developer has patented that one way, the developer can exercise its patent rights to stop unlicensed uses.[219]

D. A Declining Case for Copyright?: The Trend Toward Patents After Altai *and* Sega

The "thinness" of copyright protection for programs after *Altai* and *Sega* seems to have contributed to a shift among software developers away from heavy reliance on copyright protection for program SSO and toward a greater reliance on patents.[220] Since the mid-1990s,

---

213 *Id.* at 1526 (citation omitted). I have previously argued that extending copyright protection to machine-executable forms of programs would inhibit disclosure of program contents, which would make it difficult for copyright to promote the progress of science, the constitutional goal of copyright law. *See* Samuelson, *CONTU Revisited*, *supra* note 2, at 705–27. The Ninth Circuit mitigated that risk by allowing reverse engineering for a legitimate purpose. It did not, however, recognize that a contrary ruling would make copyright in machine-executable code into a super-strong form of trade secrecy protection for programs.

214 *Sega*, 977 F.2d at 1524, 1528–32 (treating one segment of Sega code as too functional to qualify for copyright protection).

215 Pamela Samuelson, *Are Patents on Interfaces Impeding Interoperability?*, 93 MINN. L. REV. 1943, 1959 (2009).

216 Prior to *Sega*, some commentators had argued that reverse engineering of object code should be treated as both copyright infringement and trade secret misappropriation. *See, e.g.*, Allen R. Grogan, *Decompilation and Disassembly: Undoing Software Protection*, COMPUTER LAW., Feb. 1984, at 1.

217 Samuelson, *supra* note 215, at 1959.

218 *Id.*; *cf.* Oskar Liivak, *The Forgotten Originality Requirement: A Constitutional Hurdle for Gene Patents*, 87 J. PAT. & TRADEMARK OFF. SOC'Y 261, 297 (2005) (advocating adoption of the "merger" doctrine for patents).

219 Samuelson, *supra* note 215, at 1959.

220 *See* Josh Lerner & Feng Zhu, *What is the Impact of Software Patent Shifts?: Evidence from* Lotus v. Borland 26 (Nat'l Bureau of Econ. Research, Working Paper No. 11,168, 2005)

software developers have often obtained patents for program internals, such as algorithms and data structures.[221] Such patents are, however, generally more useful for defensive, rather than for offensive, purposes.[222] That is, developers tend to seek patents on such internal design elements to assure themselves the freedom to develop software embodying these innovations.[223] Further, patents may be useful for building a portfolio of intellectual property assets, which firms can trade (e.g., by cross-licensing agreements) if a competitor asserts a patent against them.[224]

Although patents on program internal designs are attractive in these defensive respects, they are often difficult to assert offensively (i.e., to stop competitors from using them) because such elements are typically difficult to discern in commercially distributed object code.[225] Because infringement is difficult to detect, patents on internal program designs are difficult to enforce. These limitations have helped to allay worries, such as those expressed in *The Uneasy Case*, that patents for program innovations would have harmful effects on competition and ongoing innovation in the software industry.[226]

(presenting evidence of surge software-innovation patents in the mid-1990s). The patentability of algorithms and other program-related inventions had been called into question in the 1970s by decisions such as *Gottschalk v. Benson*, 409 U.S. 63, 71, 73 (1972) (ruling that an algorithm for transforming binary-coded decimals to pure binary form was not patentable subject matter). In the early 1980s, however, the Supreme Court ruled that program-related inventions could be patented in *Diamond v. Diehr*, 450 U.S. 175, 192–93 (1981). The *Diehr* decision was initially perceived as a modest change in the patent landscape as to program-related inventions because: (1) the Court was deeply divided; (2) the majority opinion did not repudiate the Court's earlier rulings on the unpatentability of certain program innovations; and (3) the case involved a traditional manufacturing process (i.e., curing rubber) that included a computer-program step. *See* Samuelson, *supra* note 215, at 1960 n.79. In the 1990s, the Federal Circuit embraced a much more expansive interpretation of what could be patented. *See, e.g.*, AT&T Corp. v. Excel Commc'ns, Inc., 172 F.3d 1352, 1361 (Fed. Cir. 1990). This led to a surge in patents for software innovations that has not abated, notwithstanding a Supreme Court decision that limited patent subject matter to some degree. *See* Bilski v. Kappos, 130 S. Ct. 3218 (2010).

221 *See* Samuelson, *supra* note 215, at 1962 n.95; *see also* Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 1, 10-11 (2001) (discussing the various Federal Circuit decisions that led to the patentability of algorithms).

222 *See* Samuelson, *supra* note 215, at 1962 n.95.

223 *See id.*

224 *See id*; *see also* Gideon Parchomovsky & R. Polk Wagner, *Patent Portfolios*, 154 U. PA. L. REV. 1, 30–31 (2005). Software patents may be useful to firms for obtaining financing from venture capitalists. *See* Samuelson, *supra* note 215, at 1964 n.95; *see also* Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 TEX. L. REV. 961, 977 (2005).

225 *See* Samuelson, *supra* note 215, at 1962 n.95.

226 *See supra* notes 161–62 and accompanying text.

E. Summary of Software Copyright Developments from 1970–2000

While patents have filled in some of the gaps in the scope of copyright protection for software innovations, the case for copyright protection for computer programs became stronger in the thirty years after the publication of *The Uneasy Case*, both for systems software and for applications. The software industry grew significantly in those decades, and copyright was at least one factor contributing to this growth. As Breyer predicted, copyright came to be tailored through the common law process[227] in cases such as *Altai* and *Sega*, as courts recognized that precedents involving novels, dramatic plays, and fabric design offered little doctrinal guidance for addressing the challenges that interoperability and other functional design elements of software posed.[228] After a brief period of overprotection in the aftermath of the *Whelan* decision,[229] *Altai* corrected this problem.[230] Despite some concerns that *Altai* would lead to underprotection of computer programs,[231] the waning years of the twentieth century brought considerable growth to the software industry, which suggests that the underprotection fears expressed by some commentators have not been realized. Although software copyright law stabilized in a manner that seems to have contributed to the success of the industry, some software industry developments in the first part of the twenty-first century have made the case for software copyrights more uneasy than in the 1980s and 1990s. These developments are discussed below in Part III.

III. HAS THE CASE FOR SOFTWARE COPYRIGHTS BECOME UNEASIER IN THE TWENTY-FIRST CENTURY?

The U.S. software industry continued to grow in the first decade of the twenty-first century. According to one industry report, there were 6918 software-development firms in the United States in 2010 that were expected to generate more than $150 billion in revenues

---

227 *See* Breyer, *supra* note 1, at 290–91.

228 See Samuelson, *Why Copyright Excludes*, *supra* note 2, at 1961–73 for a more detailed discussion of the reasons why copyright protection is not—and should not be—available to functional design elements of programs, such as algorithms and macro command systems.

229 *See supra* notes 161–62 and accompanying text.

230 *See supra* notes 185–88 and accompanying text.

231 *See, e.g.*, Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559, 2561 (1994).

that year.[232] The Business Software Alliance ("BSA"), a trade industry group, has painted an even rosier picture of the U.S. software industry.[233] BSA reports that software and related service industries employed 1.7 million people in 2007 and contributed more than $261 billion to the U.S. gross domestic product that year.[234] BSA also reported that in 2008, the U.S. software firms generated $136.6 billion in revenues globally from the sale of packaged software, representing a 45.9% share of that market.[235] BSA has calculated that roughly 30% of the $297 billion in total spending for packaged software was for PC software; the other 70% was spent on server and custom software.[236] Yet another industry report predicts that the global software market will be $330 billion by 2014.[237]

Although these figures are helpful in providing an overall picture of revenues generated from software development and services, they do not necessarily support the case for copyright for computer programs. Indeed, several significant developments in the software industry raise questions about how important copyright protection now is to enabling developers to recoup their R&D investments in software. For example, because the Internet has become such a ubiquitous phenomenon, there has been a substantial rise in the development of network computing and software that take advantage of the Internet as a platform.[238]

A second important development is that software is now commonly embedded in hardware of all kinds (cars, toasters, cell phones, just to name a few).[239] As in the early days of bundled software, the

---

232 *Software Publishing in the U.S.*, IBISWORLD (Feb. 9, 2011), http://www.ibisworld.com/industry/default.aspx?indid=1239.

233 *See* BUS. SOFTWARE ALLIANCE, SOFTWARE INDUSTRY FACTS AND FIGURES 1 (2009), *available at* http://www.bsa.org/country/Public%20Policy/~/media/Files/Policy/Security/General/sw_factsfigures.ashx.

234 *Id.*

235 *Id.*

236 *Id.*

237 *Market Report*, *"Software: Global Industry Guide", published*, PR-INSIDE.COM (May 17, 2010), http://www.pr-inside.com/market-report-software-global-industry-r1894504.htm.

238 *See, e.g.*, Samuelson, *supra* note 215, at 1970 n.140. *See generally* YOCHAI BENKLER, THE WEALTH OF NETWORKS: HOW SOCIAL PRODUCTION TRANSFORMS MARKETS AND FREEDOM (2006); BARBARA VAN SCHEWICK, INTERNET ARCHITECTURE AND INNOVATION (2010); JONATHAN ZITTRAIN, THE FUTURE OF THE INTERNET—AND HOW TO STOP IT (2008).

239 *See, e.g.*, Cliff Saran, *Lessons for Software Developers from Toyota's ABS Safety Alert*, COMPUTER WKLY. (Feb. 12, 2010, 12:30 PM), http://www.computerweekly.com/Articles/2010/02/12/240286/Lessons-for-software-developers-from-Toyota39s-ABS-safety.htm; *see also* Paul Kaihla, *The Ghosts in the Machines*, CANADIAN BUS., Sept. 17, 2001, at 25, 26 (noting that "the average American encounters 150 embedded systems on any given day").

costs of developing these programs can be recouped through revenues generated by the sale of the hardware in which they are embedded.[240] Thus, it would seem that copyright is not necessary to bring these kinds of programs into existence. Estimates of the size of the software market do not generally take embedded software of this sort into account. It would, in any event, be difficult to assess the contributions that embedded software makes to the market for the hardware in which they are embedded. Yet, if one is to have a complete picture of the role that software plays in the U.S. economy today, it would be wrong to leave this important development out of the picture entirely.

A third significant development is that it appears that seventy percent of the total investment in the development of software in the United States in the early twenty-first century is either custom-developed software or software that firms develop for their internal uses.[241] As Breyer noted in *The Uneasy Case*, custom-developed software does not really need copyright protection to induce its creation. Nor do firms really need copyright protection for software they develop for their own internal uses, such as quality assurance software.

A fourth development is a substantial rise in the use and economic significance of open-source software.[242] Breyer was skeptical about the need for copyright protection because the creation of software by educators, researchers, as well as computer users themselves was not induced by the promise of copyright's exclusivity.[243] Although educators and researchers continued to develop and share free software from 1970–2000,[244] free and open-source software has become a mainstream phenomenon in the twenty-first century and is

---

240 *See supra* note 143 and accompanying text.

241 Bureau of Econ. Analysis, *Data Tables: Software Investment and Prices, by Type*, U.S. DEP'T COM., http://www.bea.gov/national/xls/soft-invest.xls (last updated Aug. 18, 2009). These data show that in 2008, only thirty percent of the total software expenditures were for the development of prepackaged software; thirty-three percent were for custom-developed software; and thirty-six percent for the development of software for internal uses. *Id.*

242 Five years ago, open-source software accounted for less than ten percent of software used by organizations, but this is expected to rise to thirty percent by mid-2012. *See* Alison Diana, *Open Source Approaching 30% of Enterprise Software*, INFORMATIONWEEK (Feb. 9, 2011), http://www.informationweek.com/news/software/enterprise_apps/showArticle.jhtml?articleID=229208752. An industry study from 2009 predicts a 22.4% compound annual growth rate in worldwide revenue for open-source software, reaching $8.1 billion by 2013. *See Open Source Software Market Accelerated by Economy and Increased Acceptance from Enterprise Buyers, IDC Finds*, BUSINESSWIRE (July 29, 2009), http://www.businesswire.com/news/home/20090729005107/en/Open-Source-Software-Market-Accelerated-Economy-Increased.

243 *See supra* note 55 and accompanying text.

244 *See supra* note 55 and accompanying text.

pervasive today.[245] The Linux operating system is one of the most widely known examples of collaboratively developed open-source software.[246] However, it is only one of many thousands of such projects.[247] Many mainstream companies, such as IBM, are contributing substantial resources in support of Linux and other open-source projects.[248] Software companies that provide open-source software to their customers generally recoup their investments through the sale of services (e.g., to install, maintain, or customize the software) or complementary assets (e.g., proprietary add-on programs that perform specialized functions).[249]

A fifth noteworthy development is the migration of some commercially significant software from a prepackaged mass-market item to a service available "in the cloud."[250] Between 1980 and 2000, consumers generally purchased software and installed it on their computers.[251] These copies generally persisted through the life of the computers on which they were installed and were likely be loaded onto the users' new computers if and when the old ones became obso-

---

245 *See supra* note 242. Although open-source software might well be developed in the absence of copyright protection, open-source developers today actually invoke copyright as a basis upon which to impose license restrictions that are often aimed at maintaining the openness of the software and preventing its capture for proprietary projects. *See, e.g.*, Brian W. Carver, Note, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 443–44 (2005).

246 *See, e.g.*, Pamela Samuelson, *IBM's Pragmatic Embrace of Open Source*, 49 COMM. ACM, Oct. 2006, at 21 (explaining why IBM is willing to commit significant resources to supporting the development of Linux).

247 *About Sourceforge*, SOURCEFORGE, http://sourceforge.net/about (last visited July 24, 2011) (maintaining a list of over 260,000 open-source projects).

248 *See* Samuelson, *supra* note 246, at 21.

249 *See* JOSH LERNER & MARK SCHANKERMAN, THE COMINGLED CODE: OPEN SOURCE AND ECONOMIC DEVELOPMENT (2010); STEVEN WEBER, THE SUCCESS OF OPEN SOURCE SOFTWARE 195–97 (2007).

250 *See, e.g.*, DELOITTE, *supra* note 11, at 30–33. Cloud computing is "a model for delivering on-demand, self-service resources with ubiquitous network access, location-independent resource pooling, rapid elasticity, and a pay per use business model." DELOITTE, CLOUD COMPUTING: STORMS ON THE HORIZON 2 (2010), *available at* http://www.johnseelybrown.com/cloudcomputingdisruption.pdf. One commentator predicts that cloud services will become a growing portion of information technology industry growth, likely to generate about one-third of the industry's net new growth in 2013. *See* Frank Gens, *IT Cloud Services Forecast—2008, 2012: A Key Driver of New Growth*, IDC EXCHANGE (Oct. 8, 2008), http://blogs.idc.com/ie/?p=224. A survey of technology experts and stakeholders found that, by 2020, most people will access applications and information primarily in the cloud, rather than on their desktops. *See* Janna Quitney Anderson & Lee Rainie, *The Future of Cloud Computing*, PEWRESEARCHCENTER (June 11, 2010), http://pewinternet.org/~/media//Files/Reports/2010/PIP_Future_of_the_Internet_cloud_computing.pdf. This is already beginning to happen. *Id.*

251 *See supra* Part II.A–B.

lete or died. In the twenty-first century, however, many software developers are making programs available as services (e.g., "give us your data, we will process it, and we will let you know what the answers are").[252] Even traditional software product companies, such as Microsoft, are making some of their software available in the cloud as a service.[253] While cloud computing is still in the early stages, some commentators predict that it will become a major sector of the software industry in the near future.[254] If no one but the developer of such software ever has access to a machine-executable form of the program, copyright protection is arguably unnecessary. Software as a service bears some resemblance to the time-sharing systems that fed Breyer's skepticism about the economic case for copyrighting software.[255]

A sixth development is that many large, successful software firms have become hybrid providers of software and services to large enterprises.[256] IBM is a good example. Although IBM still makes and sells computer hardware and software, it now makes the bulk of its income from selling services to enterprise customers.[257] In 2010, IBM's revenues totaled $99.87 billion, of which 57% percent came from the sale of services.[258] This is more than twice its revenues from software sales, and more than three times what it made from the sale of computer systems.[259] Because of the close and ongoing relationship firms like IBM have with the enterprise customers for whom they provide services, and because of the complex contractual arrangements that bind service providers and their customers, copyright plays little role in the recoupment of R&D expenses for these hybrid systems.[260]

A seventh development is the rise of commercially significant software platforms that recoup investments in programming by means

---

252 *See e.g.*, SALESFORCE, http://www.salesforce.com (last visited July 24, 2011) (Using the following advertisement to promote cloud services: "No hardware. No software. No headache. Move your business to the cloud with the trusted leader in cloud computing . . . .").

253 *Cloud Power*, MICROSOFT, http://www.microsoft.com/en-us/cloud/default.aspx?fbid=5ONpXX7dQCJ (last visited July 22, 2011).

254 DELOITTE, supra note 11, at 31 fig.30 (predicting software as a service will be a $16 billion market by 2013).

255 Breyer, *supra* note 1, at 346–47.

256 *See, e.g.*, MICHAEL CUSAMANO, THE BUSINESS OF SOFTWARE: WHAT EVERY MANAGER, PROGRAMMER, AND ENTREPRENEUR MUST KNOW TO THRIVE IN GOOD TIMES AND BAD 273–74 (2004).

257 *See* IBM, 2010 ANNUAL REPORT 25–26 (2010), *available at* ftp://public.dhe.ibm.com/annualreport/2010/2010_ibm_annual.pdf.

258 *Id.*

259 *Id*.

260 CUSAMANO, *supra* note 256, at 49–51.

other than the sale of copies of software in the marketplace. For example, Google and Facebook make substantial revenues from advertising tailored to users who come to their sites to make use of services available there. Paypal and eBay generate revenues through taking a cut on transactions enabled by their platforms. Many software platforms—YouTube being one prominent example—attract those who develop and consume user-generated content, the overwhelming majority of which is both produced and disseminated via these platforms without the incentives provided by copyright protection.

An eighth development is the rise of technical protection measures ("TPMs") and mass-market licenses to regulate the use of software products. TPMs first appeared in mass-market software in the late 1980s, but at that time, they proved unpopular with many customers and were competed away.[261] However, many software products now use more sophisticated TPMs, which, pursuant to new legislation, are illegal to bypass.[262] As a result, TPMs provide a powerful means for protecting computer programs, and have therefore lessened the need to rely on copyright protection. The software industry also makes wide use of mass-market licenses, as it has for decades, although the enforceability of these licenses has been the subject of considerable controversy and mixed outcomes in judicial decisions.[263] Still, copyright may play a less important role in protecting software products when these adjunct forms of protection are employed. Breyer might be inclined to agree if he reassessed the case for copyright for computer programs in the current era.

Finally, there is also some evidence that legal protections are actually less important to software developers than intellectual property professionals may think. A recent survey of software entrepreneurs shows that these entrepreneurs do not perceive copyright to be very

---

261 *See, e.g.*, Julie E. Cohen, Lochner *in Cyberspace: The New Economic Orthodoxy of "Rights Management,"* 97 MICH. L. REV. 462, 521 n.221 (1998) (noting that consumers were willing to pay a higher price for unprotected software). *See generally* COMPUTER SCI. & TELECOMMS. BD., NAT'L RESEARCH COUNCIL, THE DIGITAL DILEMMA: INTELLECTUAL PROPERTY IN THE INFORMATION AGE 152–76, 282–303 (2d prtg. 2000) (discussing technical protection measures and their uses to protect software and other forms of copyrighted content).

262 *See* 17 U.S.C. § 1201 (2006) (proscribing circumvention of technical protection measures used by copyright owners to protect access to or certain uses of copyrighted content).

263 *See* Pamela Samuelson & Kurt Opsahl, *Licensing Information in the Global Information Market: Freedom of Contract Meets Public Policy*, 21 EUR. INTELL. PROP. REV. 386, 387–88 (1999) (discussing competing views on the enforceability of mass-market licenses). One recent Ninth Circuit decision has upheld the enforceability of a shrink-wrap license restriction on transfers (i.e., resales) of copies of licensed software. *See* Vernor v. Autodesk, Inc., 621 F.3d 1102, 1113–14 (9th Cir. 2010).

important to their firms' ability to attain competitive advantage in the marketplace.[264] Far more important to this key objective is first-mover advantage.[265] A first mover has the chance to gain an important advantage in the market over other players because network effects may set in before second comers are able to develop a competing product that will lure customers away from the first mover's orbit.[266] Complementary assets are not quite as important to software entrepreneurs as first-mover advantage, but they too are perceived as more important to success in the market than any form of intellectual property protection.[267] Finally, entrepreneurs perceive copyright as moderately important to software entrepreneurs, but only slightly more important than trademark and secrecy protections.[268]

CONCLUSION

Breyer's careful empirical and economic assessment of the software industry in *The Uneasy Case* provided inspiration for this Article's historical review of forty years of software industry development. Breyer was right that the software industry in 1970 did not really need copyright protection as a means to induce investment in software R&D. He was also right that the rise of the market for general-purpose applications that could be sold to a large customer base for low prices in subsequent years made the case for copyright protection stronger. He did not foresee that the case for copyrighting programs might weaken again after having gotten stronger, though this Article suggests it has.

By observing that the economic case for copyrighting programs is uneasier now than in the 1990s, I do not mean to suggest that copyright protection for computer programs should be repealed. At this

---

264 Stuart J.H. Graham et al., *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L. J. 1255, 1290 fig.1 (2009).

265 *Id*.

266 *Id*. at 1296–97.

267 *Id*. at 1290 fig.1.

268 *Id*. Software entrepreneurs ranked patents as the least important means of attaining a competitive advantage. *Id*. Yet, it should be noted that software firms have a strong interest in copyright protection insofar as mass-marketed software is infringed by users. *See* BUS. SOFTWARE ALLIANCE, SIXTH ANNUAL BSA-IDC GLOBAL SOFTWARE '08 PIRACY STUDY 1 (2009), *available at* http://portal.bsa.org/globalpiracy2008/studies/globalpiracy2008.pdf (estimating that software companies suffered $53 billion in lost sales because prospective customers copied software instead of buying it). Although Breyer doubted that firms that simply copied another firm's software would be able to do so profitably, *see* Breyer, *supra* note 1, at 344–45, he did not anticipate that users would engage in infringing acts that might, if unchecked, undermine incentives to invest in software development.

juncture, legal protection for computer programs through copyright law is an international norm, and copyright undoubtedly plays some role in inducing software producers to invest in software development and in enabling them to recoup those investments. Copyright protection has, in fact, become so deeply entrenched in software protection law and in software industry's expectations that it will be with us and the software industry for decades to come, regardless of whether it really is (or is not) economically necessary.