A BRIEF HISTORY OF SOFTWARE PATENTS (AND WHY THEY'RE VALID)

Adam Mossoff*

I. Introduction

Today, there is significant debate over whether computer programs should be protected by the patent system. Although some scholars and lawyers argue about how best to apply the specific legal requirements in assessing the patentability of these inventions, such as determining their novelty or nonobviousness,[1] there is a more highly charged legal and public policy debate as to whether these inventions should be patentable at all. In patent law parlance, does software fall within the scope of patentable subject matter defined by § 101 of the Patent Act?[2]

Unfortunately, the debates about "software patents" are rife with extensive confusion and misinformation, about both the law and the technology. In recent years, the Court of Appeals for the Federal Circuit has become deeply confused about this issue, reaching a nadir with its highly fractured 2013 en banc decision in *CLS Bank v. Alice* Corp.[3] Its one-paragraph per curiam opinion invalidating Alice Corporation's patent as an abstract idea and thus unpatentable under § 101 was accompanied by 135 pages of concurring and dissenting opinions, none of which garnered a majority. Commentators were highly critical of the *CLS Bank* decision,[4] and, unsurprisingly, the Supreme Court granted cert in what became the now-styled *Alice Corp. v. CLS Bank.*

The *Alice* Court's opinion clocked in at only 17 pages,[5] and while this brief and surprisingly unanimous opinion by Justice Clarence Thomas was eminently more readable than the Federal Circuit's sprawling mess, it did not settle the debate. In response to the Federal Circuit's breakdown on the patentability of computer programs, the Court's cert grant framed the issue very broadly: "Whether claims to computer-implemented inventions—including claims to systems and machines, processes, and items of manufacture—are directed to patent-eligible subject matter within the meaning

* Professor of Law, George Mason University School of Law. For comments, thank you to Matthew Barblan, Ron Katznelson, Michael Risch, and Robert Sachs. Thank you also to Steven Tjoe and Wen Xie for their research assistance.

[1] *See* 35 U.S.C. §§ 102-103.

[2] *See* 35 U.S.C. § 101 ("Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.").

[3] 717 F. 3d 1269 (2013) (en banc), *aff'd,* Alice Corp. Pty Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014).

[4] *See* John Kong, *The Alice in Wonderland En Banc Decision by the Federal Circuit in CLS Bank v. Alice Corp*, IPWATCHDOG (May 14, 2013, 3:16 pm), http://www.ipwatchdog.com/2013/05/14/the-alice-in-wonderland-en-banc-decision-by-the-federal-circuit-in-cls-bank-v-alice-corp/id=40344/.

[5] *See* Alice Corp. Pty Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014).

of 35 U.S.C. § 101 as interpreted by this Court?"[6] Instead of answering this broad question, however, the *Alice* Court narrowly limited its holding to addressing only the specific patent at issue and whether its "claims are patent eligible."[7] It held that they were not because it deemed the patent to cover an "abstract idea."[8] Notably absent from *Alice* was the phrase "software patent," and lawyers in the innovation industries are now arguing whether the *Alice* Court impliedly killed off computer-program patents or validated them.[9]

This essay addresses the patentability of software patents, but it takes a different tack from the increasingly substantial and seemingly "elusive" debates about what makes something "abstract" and thus unpatentable under § 101.[10] As one scholar recently observed, the difficulties in these debates "could be related to the historical path patent eligibility jurisprudence has taken."[11] This essay takes up this idea about the value of offering some historical perspective, but not in terms of the evolution of the legal doctrine and how to frame or apply a workable test. Rather, it addresses a gap in the scholarship on the evolution of computer technology and how intellectual property (IP) law has played a role in this technological development.

Given the widespread confusing rhetoric and the concomitant doctrinal upheaval, a fresh historical perspective on the *technology* is illuminating for at least two reasons. First, knowing the historical evolution of software patents—even in classic "potted history" form[12]—is valuable because it reveals that the complaints today about IP

[6] The term "computer-implemented invention" is the more technically and legally precise term for "software patent" that dominates the policy debates. *See infra* Part II.

[7] *Alice*, 134 S. Ct. at 2352.

[8] *Id.* ("We hold that the claims at issue are drawn to the abstract idea of intermediated settlement, and that merely requiring generic computer implementation fails to transform that abstract idea into a patent-eligible invention.").

[9] *Compare* Horacio Gutierrez, *Why the CLS Bank Case Matters*, MICROSOFT ON THE ISSUES, June 20, 2014, *at* http://blogs.microsoft.com/on-the-issues/2014/06/20/why-the-cls-bank-case-matters/ ("Software patents are no different than other technological or industrial inventions that are patent-eligible under Section 101. . . . The *Alice* decision is an affirmation that these innovations are patent-eligible.") *with* Gene Quinn, *SCOTUS Rules Alice Software Claims Patent Ineligible*, IPWATCHDOG, June 20, 2014, *at* http://www.ipwatchdog.com/2014/06/19/scotus-rules-alice-software-claims-patent-ineligible/id=50120/ ("Software claims as they have typically been writing now seems to result in patent ineligible claims . . . . What this means is that companies like Apple, IBM, Microsoft, Google and others have had the value of their patent portfolios nearly completely erased today.").

[10] Brief of International Business Machines as Amicus Curiae, at 12, Alice Corp. Pty Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014) ("Courts and commentators alike have repeatedly noted the elusive nature of the abstract idea doctrine."). *See also* MySpace, Inc. v. Graphone Corp. 672 F.3d 1250, 1259 (Fed. Cir. 2012) (describing the abstract idea doctrine as a "murky morass"); Donald S. Chisum, *Weeds and Seeds in the Supreme Court's Business Method Patents Decisions: new Directions for Regulating Patent Scope*, 15 Lewis & Clark 11, 14, (2011) (stating that the "abstract idea preemption inquiry can lead to subjectively-derived, arbitrary and unpredictable results").

[11] Kristen Osenga, *Debugging Software's Schemas*, __ GEO. WASH. L. REV. 25, 33 (2014).

[12] *See* MacMillan Dictionary, http://www.macmillandictionary.com/dictionary/british/potted (as of Sep. 16, 2013, 10:21 PM GMT).

protection for computer programs are nothing new. Opposition to IP protection for computer programs has long existed—predating the Federal Circuit's 1998 ruling that business methods are patentable,[13] predating the Federal Circuit's 1994 ruling that computer programs are patentable as the equivalent of a digital "machine,"[14] and predating the Supreme Court's 1980 decision that a computer program running a rubber vulcanization process was patentable.[15] In fact, computer programmers and some scholars opposed extending *copyright protection* to computer programs, as will be discussed in Part Three. This suggests that opposition to the patentability of computer programs is not rooted in any particular facts today about software or the nature of the high-tech industry.

Second, this history reveals that the shift in legal protection from copyright law in the 1980s to patent law in the 1990s was not a result of either rent-seeking by commercial firms who exploited their access to the halls of power in Congress or a reflexively pro-patent bias of the Federal Circuit. To the contrary, the historical evolution from copyright to patent law represented a natural legal progression as the technology itself evolved from the 1960s up to the mid-1990s. As it happens in our common law system—precisely because it is designed to develop this way—legal doctrines evolve in their applications in response to innovative changes in both technology and commerce. For many IP scholars who teach or write articles about Internet Law, this is all but an obvious truism.[16] The historical and technological evolution of computer programs in the high-tech industry suggests that the extension of patent protection to these inventions in the 1990s was a legitimate response in securing new innovation—precisely what the patent system is supposed to achieve in promoting "the Progress of . . . useful Arts."[17]

In three parts, this essay will survey the history of the evolution of both computer programs and "software patents." First, as a preliminary matter, it will address the confusing "software patent" rhetoric, which is necessary if only because this term now dominates the patent policy debates about computer-implemented inventions, and it unfortunately obfuscates the technological and legal facts. Second, it will discuss the digital revolution in the mid-twentieth century and the heated controversy in the 1960s and 1970s about whether software was copyrightable. Third, it will discuss the personal computer (PC) revolution in the 1980s and explain how understanding this historical development is necessary to understanding the development of patent protection for computer-implemented inventions in the 1990s.

---

[13] *State St. Bank & Trust Co. v. Signature Financial Grp.*, 149 F. 3d 1368 (Fed. Cir. 1998).

[14] *In re Alappat*, 33 F. 3d 1526 (Fed. Cir. 1994) (en banc).

[15] *Diamond v. Diehr*, 450 U.S. 175 (1981).

[16] *See, e.g.*, Metro-Goldwyn-Mayer Studies, Inc. v. Grokster, Ltd., 545 U.S. 913 (2005) (extending secondary liability doctrine for copyright to digital intermediary); Kremen v. Cohen, 337 F.3d 1024 (9th Cir. 2003) (extending common law concept of property to domain names); 47 U.S.C. § 230 (resolving split between courts as to how to apply hoary common law concepts of publisher and distributor to the Internet for purpose of applying liability tests for defamation and other legal duties).

[17] U.S. Const. Art. I § 8 Cl. 8.

It bears emphasizing that this is a "potted history" (in a non-pejorative sense). In a short essay, one cannot recount every historical detail, and some historical developments are compressed into a slightly simplified retelling. Of course, one should consult more detailed historical accounts of the digital revolution and its follow-on revolutions in PCs, the Internet and the other modern high-tech marvels of the twenty-first century.[18]

II. What is a "Software Patent"?

Before we can address the history, though, it is necessary to get clear on what exactly we mean by a "software patent." This is necessary because this term is not a term of art in patent law.[19] It is in fact an odd moniker. Aside from the similarly mislabeled debate over "DNA patents,"[20] nowhere else in the patent system do we refer to patents on machines or processes[21] in a specific technological field in this way; for instance, people do not talk about "automobile brake patents"[22] or "sex toy patents"[23] as their own category of patents deserving of approval or scorn. Even worse, as it used in the public policy debates, this term suffers from both definitional problems and subject matter problems, as will be described here.

One of the primary problems with the term "software patent" is that, like other widely used terms in the patent policy debates today,[24] it lacks an objective definition.[25] For instance, many critics argue that "software patents" are patents on "mathematics"[26] or patents on a "mathematical algorithm,"[27] but this is sophistry. As commentators have

---

[18] *See, e.g.*, T.R. REID, THE CHIP (2001) (recounting the scientific and technological developments that made the Digital Revolution possible).

[19] *See* Osenga, *supra* note 11, at 29 (observing that the PTO "does not have a specific classification for 'software' patents").

[20] *See* Adam Mossoff, *A Century-Old Form of Patent*, N.Y. TIMES, Jun. 6, 2013, http://www.nytimes.com/roomfordebate/2013/06/06/can-the-human-blueprint-have-owners/a-century-old-form-of-patent.

[21] *See* 35 U.S.C. § 101 (providing that "any new and useful process, machine, manufacture, or composition of matter" is patentable).

[22] *See* KSR Int'l Co. v. Teleflex Inc., 550 U.S. 398 (2007).

[23] *See* Ritchie v. Vast Resources, Inc. (d/b/a Topco Sales), 563 F.3d 1334 (Fed. Cir. 2009) (Posner, J.).

[24] *See* Adam Mossoff, *The SHIELD Act: When Bad Economic Studies Make Bad Laws*, Center for the Protection of Intellectual Property Blog (Mar. 15, 2013), http://cpip.gmu.edu/2013/03/15/the-shield-act-when-bad-economic-studies-make-bad-laws/ (identifying how "patent troll" lacks any definition and is used non-objectively in patent policy debates).

[25] *See* Stuart J.H. Graham & David C. Mowery, *Software Patents: Good News or Bad News?*, *in* INTELLECTUAL PROPERTY RIGHTS IN FRONTIER INDUSTRIES: SOFTWARE AND BIOTECHNOLOGY 74 (Robert W. Hahn, ed., 2005) (observing that "no widely accepted definition of *software patent* exists").

[26] *See* End Soft Patents, *Software is math* (as of Sep. 19, 2013), http://en.swpat.org/wiki/Software_is_math.

[27] This characterization of computer programs as merely "mathematical algorithms" is an unfortunate byproduct of the Supreme Court's decision in *Gottschalk v. Benson*, 409 U.S. 63 (1972), in which Justice William O. Douglas described an invention of a fundamental software program for running

repeatedly recognized, a word processing program like Word for Windows, an email client like Thunderbird, or a data encryption program like Folder Lock are not the same thing as 2+2=4,[28] and the fact that computer programs use mathematics is an argument that proves too much. All patented innovation uses mathematics; in fact, physicists love to say that the universal language of the universe is mathematics.[29] If taken seriously, the argument that a "web browser, spreadsheet, or video game *is* just math and therefore it's not … eligible for patent protection,"[30] would invalidate all patents if applied equally to other inventions, especially processes and methods. All inventions of practically applied processes and machines are reducible to mathematical abstractions and algorithms; for example, a patentable method for operating a combustion engine is really just an application of the law of PV=nRT, the principles of thermodynamics, and other laws of nature comprising the principles of engineering. As the *Alice* Court recognized, "[a]t some level, 'all inventions . . . embody, use, reflect, rest upon, or apply laws of nature, natural phenomena, or abstract ideas.'"[31]

Complicating things even further, the term "software patent," even when it is not being used in a way that invalidates all patents, is often used to refer to many different types of patented innovation in different technological fields of art. Software is ubiquitous today. It is used in automobiles, coffee machines, and refrigerators—not just in laptop computers or on the servers that make up the Internet.[32] Accordingly, the term has been used to encompass all inventions that use some type of computer software

---

all computers as an "algorithm." *Id.* at 65 ("A procedure for solving a given type of mathematical problem is known as an 'algorithm.' The procedures set forth in the present [patent] claims are of that kind."). Justice Douglas thus concluded that the invented computer program was an unpatentable abstract idea:

> It is conceded that one may not patent an idea. … The mathematical formula involved here has no substantial practical application except in connection with a digital computer, which means that if the judgment below is affirmed, the patent would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself.

Id. at 71-72. This was an unfortunate misinterpretation of the nature of computer programs as such, and it has caused much confusion in patent law about both computer programs and what makes them patentable inventions. What is notable, as is made clear in this essay, is that this confusion about the nature of computer programs in 1972 was perhaps understandable, if only because the PC Revolution had not yet occurred and thus it was much harder for judges to understand what made computer programs valuable as separate (patentable) inventions from the computer hardware on which they ran.

[28] *See* Gene Quinn, *Groklaw Response: Computer Software is Not Math*, IPWatchdog (Dec. 15, 2008, 6:30 am), http://www.ipwatchdog.com/2008/12/15/computer-software-is-not-math/.

[29] *See* Carolyn Y. Johnson, *A talk with Mario Livio – Is Mathematics the Language of the Universe*, Boston Globe, Feb. 8, 2009, http://www.boston.com/bostonglobe/ideas/articles/2009/02/08/a_talk_with_mario_livio/.

[30] Timothy B. Lee, *Software is Just Math*, Forbes, Aug. 11, 2011, http:/www.forbes.com/sites/timothylee/2011/08/11/software-is-just-math-really/.

[31] *Alice*, 134 S. Ct. at 2354 (quoting Mayo Collab. Serv. v. Prometheus Lab., Inc., 132 S. Ct. 1289, 1293 (2012).

[32] *See* Osenga, *supra* note 11, at 30-31 (observing that software is in hybrid cars, washing machines, cell phones, etc.).

program in their implementation. For example, the GAO Report on Patent Litigation (August 2013) claims that "[b]y 2011, patents related to software made up more than half of all issued patents."[33] This rather surprising assertion only makes sense if one includes not just classic computer programs among total issued patents, but *all* inventions that require some type of software program regardless of whether the invention comprises a software program itself or merely uses a software program to implement it.[34]

For ease of reference given the ubiquity of this term in the policy debates, this essay uses "software patents," but it is limited solely to patents on a set of machine-readable instructions that direct a central processing unit (CPU) to perform specific operations in a computer.[35] In short, "software" means a *computer program*, such as a word processing program (e.g., Word), a spreadsheet (e.g., Excel), or even programs run on computers on the Internet, such as Google's search algorithm, Facebook, eBay, etc. Of course, the reality is far more complicated than this, but that's not the point of this essay nor is it necessary to explore these complexities to prove its historical thesis.

In fact, few people realize the vast numbers of valid and valuable patents on computer programs. The entire Internet rests on patented innovation in computer programs: the packet-switching technology used to transmit information over the Internet was patented by Donald Watts Davies.[36] Robert Kahn and Vinton Cerf, the inventors of the TCP/IP packet-switching protocol, patented their invention of a packet-switching version of a knowbot.[37] Larry Page and Sergy Brin patented their famous search algorithm when they were graduate students at Stanford, and they raised venture capital funding for their start-up company, Google.[38] There are slews of other valid patents on

---

[33] U.S. Gov't Accountability Office, GAO-13-465, INTELLECTUAL PROPERTY: Assessing Factors That Affect Patent Infringement Litigation Could Help Improve Patent Quality 21 (2013), http://www.gao.gov/products/GAO-13-465.

[34] *See Patent Statistics in the GAO Report*, HIGH TECH INTELLECTUAL PROPERTY LEGAL BLOG (Sep. 18, 2013), http://blog.hiplegal.com/2013/09/gao_softwarepatents/("The problem, of course, is that there are no 'exclusively software' classes. So if an entire patent class is counted, it is extremely likely to include non-software cases as well.").

Hal Wegner and others also claim that the GAO actually made an outright error in its counting methodology. *See* Hal Wegner, *GAO Patent Litigation Report (con'd): "[P]atents related to software ma[ke] up more than half of all … patents,"* LAIPLA (Aug. 2, 2013), http://www.laipla.net/gao-patent-litigation-report-cond-patents-related-to-software-make-up-more-than-half-of-all-patents/ ("The GAO authors apparently counted 20,000 software patents instead of 2,000 under the methodology at p.12 n.27 (explaining Figure 1). Thanks to Greg Aharonian for sharing this information with the patent community."); *see also* http://www.global-patent-quality.com/GRAPHS/SoftElec.htm (reporting Greg Aharonian's statistics on issued patents that show that even the 2,000 number is almost twice the actual rate of issuance of "software" patents).

[35] *See* Wikipedia, *Software* (as of Sep. 19, 2013), http://en.wikipedia.org/wiki/Software.

[36] *See* Patent No. 4,799,258.

[37] *See* Patent No. 6,574,628. A knowbot was an early version of a type of search engine. *See* John Markoff, *Creating a Giant Computer Highway*, N.Y. Times (1990), http://www.nytimes.com/1990/09/02/business/creating-a-giant-computer-highway.html.

[38] *See* Patent No. 6,285,999. There are several patents, but is one of the core ones.

technologically and commercially valuable computer programs, such as an early one from 1993 for one of Excel's core spreadsheet functions.[39]

To understand why these and many other patents on computer programs are both valuable and valid, it is necessary to understand whence computer programs came, how they changed in both their technological and commercial function after the 1970s, and why patent law was extended to secure this technological innovation in the early 1990s.

III. The Digital Revolution and the Copyright Controversy

The progenitor of software patents is found in the early years of the Digital Revolution with the invention of the integrated circuit in 1958-1959 (independently invented by Jack Kilby and Robert Noyce).[40] At that time, "software," at least as we now understand this term, did not mean what we think this term means today. Software was designed for specific computers and only for those computers. To wit, what worked on a IBM mainframe did not work on a DEC minicomputer (which was the size of a refrigerator).[41]

Despite the start of the Digital Revolution a mere sixty years ago, its early growing pains have become the equivalent of "ancient history." For this reason, many people no longer remember that the protection of computer programs under copyright — something accepted today as an allegedly "obvious" legal alternative to patent protection — was originally disputed rigorously by programmers and others. The question of whether computer programs were copyrightable was a tremendous flashpoint of controversy for much of the 1960s and 1970s, which is ironic given that people today blithely assert that we don't need patent protection for computer programs because "copyright protection … makes patent protection mostly superfluous."[42] (This claim is also false, as the historical development makes clear and as will be explained shortly.)

---

[39] *See* Patent No. 5,272,628.

[40] *See* REID, *supra* note 18, at 76-80 & 91-95.

[41] Someone might ask, "Who or What is DEC?," and this is an excellent question that highlights the dynamic innovation that has been the hallmark of the high-tech industry for the past fifty years. The Digital Equipment Corporation (DEC) was one of the early high-tech firms manufacturing computers in the 1960s, ultimately bringing in multi-billion dollar revenues. *See* Wikipedia, *Digital Equipment Corporation* (as of Sep. 19, 2013), *at* http://en.wikipedia.org/wiki/Digital_Equipment_Corporation. Its founder and CEO, Ken Olson, was admired by a young Bill Gates, who wrote of Olson: "An inventor, scientist, and entrepreneur, Ken Olsen is one of the true pioneers of the computing industry. He was also a major influence in my life and his influence is still important at Microsoft through all the engineers who trained at Digital and have come here to make great software products." Chloe Albanesius, *Computing Pioneer Ken Olson Dead at 84*, PC Magazine, Feb. 8, 2011. Olson is known today for his infamous proclamation in 1977: "There is no reason for any individual to have a computer in his home." *See* Joelle Tessler, *Kenneth Olsen, Pioneering Founder of Computer Company, Dies at 84*, Wash. Post, Feb. 9, 2011, http://www.washingtonpost.com/wp-dyn/content/article/2011/02/09/AR2011020906305.html. This belief is why DEC is no longer around and why young people today no longer remember this company.

[42] Timothy B. Lee, *The Supreme Court Should Invalidate Software Patents*, FORBES, Jul. 28, 2011, http://www.forbes.com/sites/timothylee/2011/07/28/the-supreme-court-should-invalidate-software-patents/.

Despite substantial controversy, in 1964 the Registrar of Copyrights started to register copyright protection for software code for computer programs.[43] Although there was no direct legal challenge to the Copyright Registrar's decision to begin registering copyrights for computer programs, the public policy debates did not go away.[44] The controversy continued for almost two decades, especially in the courts,[45] and it was resolved only by congressional fiat with the Computer Software Copyright Act of 1980,[46] specifically authorizing the protection of software code by the Registrar of Copyrights under the Copyright Act. In sum, opposition to IP protection for computer programs has existed from time immemorial, regardless of whether it was copyright or patent.

IV. The PC Revolution and the Birth of Software Patents

It is significant that the Computer Software Copyright Act was enacted in the early 1980s because it was during this time—the late 1970s and early 1980s—that the personal computer (PC) Revolution began. This is the point in time that marks the shift away from hardware and software as a unified, single product both technologically and commercially, to hardware and software as distinct products. This is the revolution brought to us by the young hackers and computer geeks of the 1970s—Steve Jobs, Steve Wozniak, Bill Gates, Nathan Myrhvold, etc.—who conceived, designed, and implemented the idea of an operating system (OS) running on a general purpose central processing unit (CPU) that serves as the operational platform for any computer program written by anyone for performing any tasks. The first programs written and implemented by end-users in the 1970s were simple, such as playing tic tac toe or blinking lights on a circuit board in a certain pattern, but within a few scant years, more sophisticated programs began to be written and sold in the marketplace, such as word processing, spreadsheet, and computer-assisted design (CAD) programs.

For the purpose of understanding the evolution of software patents, the importance of the PC Revolution is that computer programs now became *separate products* that consumers could purchase, install, and use on their PCs (either an "IBM Compatible" or a Mac). In fact, computer programs came in a *box* that consumers

---

[43] *See* National Commission on New Technological Uses of Copyrighted Works, Final Report 82 (1979); Copyright Office Circular 31D (Jan. 1965).

[44] *See, e.g.*, Allen W. Puckett, *The Limits of Copyright and Patent Protection for Computer Programs*, 16 COPYRIGHT L. SYMP. 81, 104-05 (1968) (recognizing that there is limited copyright protection for some aspects of computer programs but that "[s]ource programs embodied in punch cards or magnetic tape present a doubtful case"); Pauline Wittenberg, Note, *Computer Software: Beyond the Limits of Existing Proprietary Protection Policy*, 40 BROOKLYN L. REV. 116, 117-18 (1973) ("With respect to computer software, such questions [about patent or copyright protection] have been under discussion in both legal and trade journals and in the courts for nearly a decade; no clear answers have emerged.").

[45] *Compare* Data Cash Systems, Inc. v. JS&A Group, Inc., 480 F. Supp. 1063 (N.D. Ill. 1979) (holding object code is not copyrightable) *and* Tandy Corp. v. Personal Micro Computers, Inc., 524 F.Supp. 171 (N.D. Cal. 1981) (holding object code in ROM is copyrightable). *See also* Synercom Tech., Inc. v. Univ. Comp. Co., 462 F. Supp. 1003, 1014 (N.D. Tex. 1978) (holding software code "formats" is not copyrightable).

[46] Pub. L. No. 96-517, 94 Stat. 3015, 3028 (1980).

physically took off shelves and purchased at checkout registers at retail stores, such as at an Egghead Software outlet, whose brick-and-mortar stores by the late 1990s went the way of DEC.[47]

The significance of a computer program becoming a separate product is that the *value* in software, what the consumer was seeking in purchasing it from the retailer, was the *function* of the program as experienced by the consumer (or "end user" in high-tech parlance). For instance, it was the value in the ease of use of a graphical user interface (GUI) of a particular word processing program, such as Word for Windows, that made it more appealing to consumers than the text-based commands of older word processing programs, such as WordPerfect. Or it was the pull-down menu in a Lotus1-2-3, the first widely successful spreadsheet program. The end user now had a word processing program with many functions in it, such as editing text, italicizing text, "cutting" and "pasting," changing margins for block quotes, etc. *This* was the value in the product sold to the consumers, and thus *this function* is what designers of computer programs competed over for customers in the marketplace. For instance, few people today remember the battle in the late 1980s and early 1990s between WordPerfect (a text-based word processor developed for the text-based command system of DOS) and Word (a pull-down menu and button-based "point and click" GUI word processor for the Apple and Windows GUI OS).

This is a key fact about computer programs that is essential to understanding the shift from copyright protection to patent protection, but it is often missed or ignored in the public policy debates about software patents. Yet, this is a widely recognized fact by many innovators who have worked for decades in the high-tech industry, or at least by those innovators who made possible the PC Revolution. Nathan Myhrvold, former executive at Microsoft, recounts how many people working in the high-tech industry in the 1980s were skeptical that a company whose sole product was software could succeed. In 1987, Myhrvold attended a

> big industry conference in the PC industry. And there was a panel discussion I participated in—"Can Microsoft Make it Without Hardware?" I swear. Now, we had a proposition and the proposition was that not only can you make software valuable without hardware; software was actually a better business without hardware, because if you separated yourself off and you just became a software company you could focus on making the software best . . . . An independent software company can target everybody's stuff.[48]

[47] *See* supra note 41 (describing the rise and fall of DEC). Egghead Software closed all its retail stores in 1998 due to the dominance of the Internet as a medium over which to order DVDs, and, eventually, through which end-users now directly purchase and download in 30 seconds their new software products or apps. *See* Wikipedia, *Egghead Software* (as of Sep. 19, 2013), http://en.wikipedia.org/wiki/Egghead_Software.

[48] Nathan Mryhvold, *Invention: The Next Software*, Intellectual Ventures, at 5 (2006), http://www.intellectualventures.com/assets_docs/Invention_Next_Software_Transcript_2006_Speech.pdf.

What Myhrvold means by "target[ing] everybody's stuff" is that a company like Microsoft could succeed in selling computer programs that provided functional value to a vast array of end users using different PCs, regardless of the manufacturer or OS. Thus, for instance, Robert Sachs, a patent attorney who specializes in high-tech innovation and serves as an evaluator for high-tech standards, explains that the "vast majority of value in software comes not from some deeply embedded algorithm that can be protected by trade secret. Rather, it comes from the creation of new functionality that has immediate and apparent value to the end user, whether that's a consumer or an enterprise."[49]

Microsoft proved the naysayers wrong, and it flourished as part of the PC Revolution wrought by its founder, Bill Gates, and by others, such as Steve Jobs and his innovative company, Apple. Yet, in the late 1980s and early 1990s, this development in new technology and new commercial intermediaries in delivering new computer programs to consumers created a problem: any programmer can replicate the GUI or other features of a commercially successful computer program—copying the valuable function of the program—without copying the literal software code that created this valuable function. In sum, the code becomes distinct from the end-user interface or the function of the program itself.

And there's the rub (to paraphrase the Bard): copyright protects someone only against copying of their literal words, not the broader idea or function represented by those words. In copyright law, this is the well-known legal rule referred to as the idea/expression dichotomy.[50] It is also reflected in the equally hoary legal rule that copyright does not protect utilitarian designs.[51]

This issue was brought to a head in the famous copyright case, *Lotus v. Borland*.[52] Lotus, the creator of the very famous spreadsheet program Lotus 1-2-3, sued Borland in 1990 for copying Lotus's innovative pull-down menus in Borland's spreadsheet program, Quattro Pro. Lotus's design of the pull-down menus in Lotus 1-2-3—these are now standard in all GUI-based computer programs—made it very efficient to use and this was a major reason for its commercial success.

---

[49] Robert R. Sachs, *Applying Can Openers to Real World Problems: The Failure of Economic Analysis Applied to Software Patents*, BILSKI BLOG (Aug. 13, 2013), http://www.bilskiblog.com/blog/2013/08/applying-can-openers-to-real-world-problems-the-failure-of-economic-analysis-applied-to-software-pat.html.

[50] *See* Baker v. Seldon, 101 U.S. 99, 104 (1879) ("[T]he teachings of science and the rules and methods of useful art have their final end in application and use; and this application and use are what the public derive from the publication of a book which teaches them. But as embodied and taught in a literary composition or book, their essence consists only in their statement. This alone is what is secured by the copyright."); Morrisey v. Proctor & Gamble Co., 379 F.2d 675, 678 (1st Cir. 1967) ("Copyright attaches to form of expression . . . .").

[51] *See Baker*, 101 U.S. at 102 ("[N]o one would contend that the copyright of the treatise would give the exclusive right to the art or manufacture described therein. … That is the province of letters-patent, not copyright. The claim to an invention or discovery of an art or manufacture … can only be secured by a patent from the government.").

[52] 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

The *Lotus* case was active for five years, and ultimately resulted in a trip to the U.S. Supreme Court, which split 4-4 in affirming the lower court (Justice Stevens recused himself), and thus the Supreme Court didn't hand down a precedential opinion.[53] As a result of the 4-4 split, though, the Court of Appeals for the First Circuit's decision in favor of Borland was affirmed by default. The First Circuit held that Lotus could not copyright its pull-down menus because these were a functional "method of operation," i.e., a utilitarian design, and not an expressive text capable of receiving copyright protection.[54] The First Circuit and the four Justices who affirmed the First Circuit were correct in applying long-standing and fundamental copyright doctrine in denying copyright protection to the *functionality* of a computer program.

By the mid-1990s, as represented in the famous *Lotus v. Borland* case, it was clear that copyright could no longer adequately secure the value that was created and sold in software programs by the fast-growing high-tech industry. The *value* in a software program is the *functionality* of the program, such as Lotus 1-2-3, Excel, WordPerfect or Word for Windows. This function was the reason why consumers purchased a program, installed it and used it on their computers, whether an Apple computer or a Windows machine. But this functionality could be replicated using myriad varieties of code that did not copy the original code, and copyright did not protect the functional components of the program that this code created for the end user—and for which the end user purchased the program in the first place.

This simple legal and commercial fact—copyright could not secure the real value represented in an innovative computer program—explains why in the mid-1990s there was a shift to the legal doctrine that could provide the proper legal protection for the innovative value in a computer program: patent law. As the Supreme Court has repeatedly recognized in contrasting patents against other IP regimes, such as copyright and trademark, "it is the province of patent law" to secure "new product designs or functions."[55]

In fact, this shift from copyright to patent law in the mid-1990s mirrors the equally important shift in the early 1980s when the courts and Congress definitively extended copyright protection to computer programs at the start of the PC Revolution. At the time, neither legal development was destined to occur by necessity, but, in retrospect, neither development was a historical accident from the perspective of the continuing success of the Digital Revolution. These two legal developments served as the fulcrums

[53] Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996).

[54] *Lotus*, 49 F.3d at 815.

[55] *Qualitex Co. v. Jacobsen Products Co., Inc.*, 514 U.S. 159, 164 (1995); *Elmer v. ICC Fabricating*, 67 F.3d 1571, 1580 (Fed. Cir. 1995) ("patent law, not trade dress law, is the principal means for providing exclusive rights in useful product features"). *See also Baker*, 101 U.S. at 102 ("[T]he exclusive right to the art or manufacture …. is the province of letters-patent, not copyright.").

by which it was possible for inventors and innovating firms, such as Apple, Microsoft, eBay, Google, etc. to commercialize these newly created values.[56]

At approximately the same time that the First Circuit and Supreme Court came to the legally correct conclusion in *Lotus v. Borland* that the functional value in the pull-down menus was not copyrightable, the Court of Appeals for the Federal Circuit expressly recognized that computer programs are patentable inventions. In its now-famous 1994 decision in *In re Alappat*,[57] the Federal Circuit ruled that a specific computer program that performed a specific and identifiable function for an end user is not an "abstract" claim to an unpatentable idea or "algorithm."[58] To the contrary, such computer programs are patentable inventions because they are a digital "machine."[59]

In essence, the Federal Circuit recognized the basic truth to which untold numbers of successful firms in the high-tech industry owe their existence: a computer program such as an Excel spreadsheet program "is not a disembodied mathematical concept which may be characterized as an 'abstract idea.'"[60] A computer program, such as Google's search algorithm, or a sub-program, such as an operation in Excel's spreadsheet, that is created, purchased and used to "perform particular functions" is the digital equivalent of "a specific machine."[61] For example, a word processing program is the equivalent in the Digital and PC Revolutions of a mechanical typewriter in the Industrial Revolution. Similarly, an e-mail produced by the functions of a word processing program in an email program, such as Outlook or Eudora, is the digital equivalent of a physical letter written by a typewriter and mailed via the U.S. Post Office to its recipient.

Given the function of the patent system in promoting and securing in the marketplace new technological innovation, the *Alappat* court was correct to recognize that the historical and technological difference between analog machines and digital machines is irrelevant with respect to the fundamental question of whether these are patentable subject matter. In fact, as any computer programmer or electrical engineer can attest, the functions of a program can be performed perfectly in either software or hardware.[62] The functional operation between the two is a distinction without a difference, except that a computer program is less costly and more efficiently sold and used by end users. There may be questions about the novelty or nonobviousness in an

[56] *See supra* notes 36-39 and accompanying text (discussing patented software that was properly secured, which made it possible for the companies that owned these patents to bring these values to the marketplace and to everyone's lives).

[57] 33 F. 3d 1526 (Fed. Cir. 1994) (en banc).

[58] *Id*. at 1545.

[59] *Id*. ("We have held that such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.").

[60] *Id.* at 1544.

[61] *Id*. at 1544-45.

[62] *See* Brief of IEEE-USA, at 8-24, Alice Corp. Pty Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014).

invention of a digital machine,[63] but its status as a new technological invention is undeniable.

In sum, the functionality of binary code in a specific computer program is in principle no different from the functionality achieved in the binary logic hardwired into computer hardware. There is no reason in logic or in patent law to assert that software is not patentable subject matter, but hardware that does the exact same thing is patentable. The fact that both types of inventions are easily identified by firms, retailers and end-users confirms that the two can be specific, real-world and useful products. This functional equivalence between hardware and software further reflects the fact that the difference between computer programs (either in software or hardware) and the mechanical machines they replaced is itself a distinction without a difference—both have been innovative inventions deserving of protection as patentable subject matter.

V. Conclusion

The American patent system has succeeded because it has secured property rights in the new innovation that has come about with each new era—and it has secured the same property rights for all types of novel, nonobvious and useful inventions. This was certainly the case in the Industrial Revolution, which produced such technological marvels as sewing machines,[64] telegraphs,[65] typewriters,[66] and telephones,[67] among many other analog inventions. It also has been the case in each phase of the Digital Revolution, as computer technology evolved from hardware to the fixed union of hardware and software to today's technological and commercial divide between hardware and software as distinct technological innovations.

As the Supreme Court recently recognized in *Bilski v. Kappos*,[68] the patentable subject matter section of the Patent Act (§ 101) is a "dynamic provision designed to encompass new and unforeseen inventions."[69] As the *Bilski* Court further recognized, a patentable subject matter test created in response to nineteenth-century (analog) innovation "may well provide a sufficient basis for evaluating processes similar to those in the Industrial Revolution — for example, inventions grounded in a physical or other tangible form. But there are reasons to doubt whether the test should be the sole criterion for determining the patentability of inventions in the Information Age."[70]

---

[63] *See* 35 U.S.C. §§ 102-103.

[64] *See* Adam Mossoff, *The Rise and Fall of the First American Patent Thicket: The Sewing Machine War of the 1850s*, 53 Ariz. L. Rev. 165 (2011).

[65] *See* Adam Mossoff, *O'Reilly v. Morse* (June 2014), *at* http://ssrn.com/abstract=2448363 (discussing invention and commercialization of the electro-magnetic telegraph).

[66] *See, e.g.*, Patent No. 79, 265 (issued June 23, 1865).

[67] *See* Patent No. 174,465 (issued Feb. 14, 1876).

[68] 130 S. Ct. 3218 (2010).

[69] *Id.* at 3227.

[70] *Id*.

The historical evolution of high-tech innovation and of the differing IP protections that arose at the critical junctures in this technological development is a testament to this basic truth. While lawyers and scholars may debate, in the words of Justice Joseph Story, the "metaphysics of the law" of the abstract idea doctrine in which the legal distinctions seem "almost evanescent,"[71] the historical and technological development of software makes clear that it is not an abstract idea. It is a twenty-first century digital machine or process. To restrict the patent system to only the valuable analog machines and processes of the nineteenth century is to turn the patent system on its head—denying today's innovators the protections of the legal system whose purpose is to promote and secure property rights in innovation.

[71] Folsom v. Marsh, 9 F. Cas. 342, 344 (C.C.D. Mass. 1841) (No. 4,901) ("Patents and copyrights approach, nearer than any other class of cases belonging to forensic discussions, to what may be called the metaphysics of the law, where the distinctions are, or at least may be, very subtle and refined, and, sometimes, almost evanescent.")