Harvard Journal of Law & Technology
Volume 29, Number 2 Spring 2016

SOFTWARE, ABSTRACTNESS, AND SOFT PHYSICALITY REQUIREMENTS

*Shane D. Anderson**

TABLE OF CONTENTS

I. INTRODUCTION

Many commentators claim the fundamental problem with software patents is their low quality, which facilitates abuse.[1] Patent trolls, which are generally described as entities that assert patent in-

* Harvard Law School, Candidate for J.D., 2016. B.Sc. in Electrical Engineering, University of Calgary, 2009. Prior to law school Shane worked for five years: one year in an undergraduate internship at Uzushio Electric Co., Ltd. in Imabari, Japan where he worked on embedded systems and developing simulators; two years at Telvent Canada in Calgary, Canada developing SCADA systems; and two years teaching English in Yoshinogari, Japan. His primary interests are in patent law, the high-tech and software industries, and the role patent law plays in promoting innovation and scientific and technological development. Shane would like to express his deep gratitude for the generous support and guidance from Professor Mark Wu and invaluable input from Professor John Golden, as well as for all of the hard work put in by Article Editor Brendon Vetter and the entire editing team at the *Harvard Journal of Law & Technology*.

1. *See, e.g.*, Jack G. Abid, *Software Patents on Both Sides of the Atlantic*, 23 J. MARSHALL J. COMPUTER & INFO. L. 815, 835 (2005); Cheryl Milone, *Stopping Abusive Patent Litigants, Not Innovation: Judicial Tools That Do No Harm*, FED. LAW., Oct.–Nov. 2013, at 40.

fringement claims but do not produce or commercialize the patented inventions, are often the ones that abuse low quality patents.[2] Scholars and legislators alike have proffered solutions to curb abuse by patent trolls.[3] These attempts, however, may simply act to soothe the symptoms of a more fundamental illness.

The patent troll problem has been characterized as stemming from software patents.[4] A study by the Government Accountability Office found that from 2007 through 2011 the number of defendants in patent lawsuits more than doubled, and software patents accounted for eighty-nine percent of the increase.[5] To combat this problem, some have advocated completely barring software patents.[6] However, such visceral reactions may stem from focusing too much on the fact that the actors abusing software patents are frequently patent trolls, which obscures the root problem that the low quality of many software patents is what makes them easy to abuse.[7]

This Note argues that the problem arises from insufficient legal guidance. The United States Patent and Trademark Office ("USPTO") should proactively address this issue by adopting soft physicality requirements for the approval of software patents. This proposal allows for the patenting of certain software, but forces the patentee to limit the scope up front by requiring a tie into essential hardware or computing platform(s). Other highly innovative nations have similarly used special industry-specific hurdles to software patenting.[8] Accordingly,

---

2. *See, e.g.*, John R. Allison, Mark A. Lemley & Joshua Walker, *Extreme Value or Trolls on Top? The Characteristics of the Most-Litigated Patents*, 158 U. PA. L. REV. 1, 24 (2009). Other definitions of what constitutes a patent troll exist. The scope of the definition largely depends on how strictly one believes that a patent holder should commercialize or practice patents held. For example, many would not consider a non-practicing university holding a patent to be a patent troll while others, essentially using "patent troll" as a synonym for "non-practicing entity," would. *See, e.g.*, Sannu K. Shrestha, *Trolls or Market-Makers? An Empirical Analysis of Nonpracticing Entities*, 110 COLUM. L. REV. 114 (2010). Others take a narrower approach and tie patent trolls to their behavior directly relating to research and development, defining them as "companies that use patents primarily to obtain license fees rather than to support the development or transfer of technology." Colleen V. Chien, *Predicting Patent Litigation*, 90 TEX. L. REV. 283, 292 (2011).

3. *See, e.g.*, Tracie L. Bryant, Note, *The America Invents Act: Slaying Trolls, Limiting Joinder*, 25 HARV. J.L. & TECH. 673, 674 (2012) (arguing that the enactment of the America Invents Act was animated at least in part to curb infringement suits by patent trolls); Abid, *supra* note 1, at 836–37.

4. *See* James Bessen, *The Patent Troll Crisis Is Really A Software Patent Crisis*, WASH. POST (Sept. 3, 2013), http://www.washingtonpost.com/blogs/the-switch/wp/2013/09/03/the-patent-troll-crisis-is-really-a-software-patent-crisis/ [https://perma.cc/5Z4S-VPL3].

5. U.S. GOV'T ACCOUNTABILITY OFF., GAO-13-465, INTELLECTUAL PROPERTY: ASSESSING FACTORS THAT AFFECT PATENT INFRINGEMENT LITIGATION COULD HELP IMPROVE PATENT QUALITY 21 (2013).

6. *See* Andrew Nieh, *Software Wars: The Patent Menace*, 55 N.Y. L. SCH. L. REV. 295, 330 (2010–2011).

7. *See infra* Part II.B.

8. For example, until October 1, 2015, patentees in Japan had to show that "information processing by software is concretely realized by using hardware resources" in order to receive a patent. *See* JAPAN PATENT OFFICE, EXAMINATION GUIDELINES FOR PATENT AND

this Note will also discuss the Japanese patent system and explore what lessons can be learned from that model.

The Note begins by discussing the history, current eligibility standards, and problems associated with software patents in Part II. In Part III, the specific proposal is developed and described. The merits of the proposal and precedential support are discussed in Part IV. Criticism is briefly discussed in Part V. Part VI concludes.

II. SOFTWARE PATENT HISTORY AND PROBLEMS

In the late 1960s, the USPTO took a hard anti-software stance, issuing examination guidelines in 1968 that held computer programs generally unpatentable.[9] A programmed computer, however, could be part of a patentable process[10] if combined with unobvious elements to produce a physical result.[11] Despite the strong directive from the USPTO on the ineligibility of software patents, the Court of Customs and Patent Appeals ("CCPA")[12] held in favor of the patentability of software.[13] With this tension, the Supreme Court intervened in a pair of landmark cases during the 1970s: *Gottschalk v. Benson* and *Parker v. Flook*.[14]

In *Benson*, the Court fell back on precedent, holding that phenomena of nature, mental processes, and abstract ideas are unpatentable[15] and finding that the claim in question was "so abstract and sweeping as to cover both known and unknown uses" of the mathematical algorithm in question.[16] A primary concern of the Court was

---

UTILITY MODEL IN JAPAN Part VII, Chapter 1, 10 (2011), http://www.jpo.go.jp/tetuzuki_e/t_tokkyo_e/Guidelines/7_1.pdf [https://perma.cc/JVQ6-KDX3] [hereinafter JPO EXAMINATION GUIDELINES].

9. *See* Diamond v. Diehr, 450 U.S. 175, 197–98 (1981) (summarizing the 1968 guidelines).

10. 35 U.S.C. § 101 defines what constitutes patentable subject material and includes "any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof." 35 U.S.C. § 101 (2012). "Process" is defined in 35 U.S.C. § 100 as, circularly, a "process, art or method, and includes a new use of a known process, machine, manufacture, composition of matter, or material." 35 U.S.C. § 100 (2012).

11. *See Diehr*, 450 U.S. at 198.

12. The CCPA was abolished in 1982 by the Federal Courts Improvement Act of 1982, which created the United States Court of Appeals for the Federal Circuit, the successor court to the CCPA.

13. *See, e.g.*, *In re* Prater, 415 F.2d 1378, 1389 (C.C.P.A. 1968) (reversing a rejection by the Patent Office Board of Appeals on a claim that involved a programmed digital computer); *In re* Bernhart, 417 F.2d 1395, 1400 (C.C.P.A. 1969) (stating that a machine "programmed in a certain new and unobvious way" is "physcially [sic] different from the machine without that program," resulting in a "new and useful improvement" of the unprogrammed machine, thus rendering it patentable subject matter under 35 U.S.C. § 101).

14. Gottschalk v. Benson, 409 U.S. 63, 64 (1972); Parker v. Flook, 437 U.S. 584, 585 (1978).

15. *Benson*, 409 U.S. at 67.

16. *Id.* at 68.

that granting such a patent would "wholly pre-empt" use of the formula and allow for its monopolization.[17]

The patent in *Flook* covered a method of updating alarm limits that relied on a mathematical formula.[18] The Court, after announcing that the case "turns entirely on the proper construction of § 101" of Title 35,[19] proceeded to treat the mathematical formula as if it "were well known" for the purposes of analysis[20] and ruled it out of the § 101 analysis.[21] The Court then found the application contained "no claim of patentable invention" because the remaining "invention," an algorithm, was directed essentially to a mathematical formula and was thus unpatentable.[22] After *Benson* and *Flook*, many doubted whether software could be patented at all.[23]

The Supreme Court reversed its position in the 1981 case *Diamond v. Diehr* in which the Court analyzed a patent on a process that utilized a computer program to cure rubber.[24] The majority differentiated *Diehr* from *Benson* and *Flook* on the grounds that the patent in *Diehr* involved a physical transformation; the Court then stated that physical transformation is "*the* clue to the patentability of a process claim that does not include particular machines."[25] In the years following *Diehr* the confusion surrounding software patents lead to repeated attempts by the courts to formulate standards and tests, often with the Supreme Court and lower courts conflicting. An example of this is the Supreme Court's denouncement of the Federal Circuit's application of the machine-or-transformation test as the sole test for patent-eligible subject matter, discussed in the next paragraph.

---

17. *Id.* at 71–72.

18. *Flook*, 437 U.S. at 585.

19. *Id.* at 588.

20. *Id.* at 592.

21. *See supra* note 10. Under the § 101 analysis, the court is to focus on the subject matter of the claimed invention itself and determine whether it falls within one of the enumerated categories of "process, machine, manufacture, or composition of matter, or any new and useful improvement thereof." 35 U.S.C. § 101 (2012).

22. *Flook*, 437 U.S. at 594–95. The patentee took issue with the Supreme Court's analysis and, as the Court recorded, "argue[d] that this approach improperly imports into § 101 the considerations of . . . §§ 102 and 103." *Id.* at 592. This is a reasonable view as the approach requires first finding the "point of novelty" in a claimed invention and then analyzing the remaining "inventive" components in isolation under § 101. This can bias the analysis, as seen in *Flook*, in that there is a diminished frame of reference for the breadth or, more importantly, the narrowness of a given patent that uses a component that is ineligible subject matter.

23. *See, e.g.*, David A. Piehler, *Patent Law – Subject-Matter Patentability – Process Patents – The Patentability of Computer Software*, 1979 WIS. L. REV. 867, 869 (1979). *But see* David A. Blumenthal & Bruce D. Riter, *Statutory or Non-Statutory?: An Analysis of the Patentability of Computer Related Inventions*, 62 J. PAT. OFF. SOC'Y 454, 518–20 (1980) (arguing that software patents may be statutorily acceptable after *Benson* and *Flook* if the patent is drafted appropriately).

24. *In re* Diehr, 602 F.2d 982, 983 (C.C.P.A. 1979).

25. Diamond v. Diehr, 450 U.S. 175, 184 (1981) (quoting Gottschalk v. Benson, 409 U.S. 63, 70 (1972)) (emphasis added).

After the Federal Circuit denounced the Freeman-Walter-Abele test (formulated in the early 1980s) as "inadequate" in *Bilski*,[26] the Federal Circuit designated a separate "machine-or-transformation test" as the determinative "test for patent-eligible subject matter."[27] The machine-or-transformation test dictates that a "process patent must either be tied to a particular machine or apparatus or must operate to change articles or materials to a 'different state or thing.'"[28] The Supreme Court quickly introduced more confusion by pulling back and announcing that "the machine-or-transformation test is a useful and important *clue*."[29] This was a nod to the role that physicality — the tying of the idea embodied by the patent's claims to the construct by which it is to be realized — plays in indicating the patent eligibility of a given invention.

After *Diehr*, the Federal Circuit continued the expansion of software patenting in its 1998 decision *State Street Bank & Trust Co. v. Signature Financial Group* by holding that a patent on a processing system implementing an investment structure was valid as a statutory machine claim.[30] The Court also supported patenting so-called "business methods,"[31] opening another scheme through which software can be patented. In the year immediately following the *State Street* decision, the number of business method patents more than doubled.[32]

A. Current Standards

The concern animating the machine-or-transformation test, as explained by the Federal Circuit in *In re Bilski*, is that a given patent may wholly preempt fundamental principles.[33] This would prevent the use of the principles in further innovation and development. The preemption concern is still prominent and is used by courts to declare

---

26. *In re* Bilski, 545 F.3d 943, 959 (Fed. Cir. 2008).

27. *Id.*

28. *Benson*, 409 U.S. at 71.

29. *See* Bilski v. Kappos, 561 U.S. 593, 604 (2010) (emphasis added).

30. *See* State St. Bank & Trust Co. v. Signature Fin. Grp., 149 F.3d 1368, 1370 (Fed. Cir. 1998).

31. *Id.* at 1375. The Supreme Court in *Bilski v. Kappos* supported the Federal Circuit and declared that "a business method is simply one kind of 'method' that is . . . eligible for patenting under § 101." 561 U.S. at 607. As is typical of the Supreme Court, it declined to accurately define what a business method is and instead relied on a definition in the pre-America Invents Act version of 35 U.S.C. § 273, which indicated that "method" means "a method of doing or conducting business." *Id.* Some have noted that prior to the *State St.* ruling the USPTO had granted patents on business methods. Kevin Schubert, *Should* State Street *Be Overruled? Continuing Controversy over Business Method Patents*, 90 J. PAT. & TRADEMARK OFF. SOC'Y 461, 462 (2008).

32. *See* Kevin M. Baird, *Business Method Patents: Chaos at the USPTO or Business as Usual?*, 2001 U. ILL. J.L. TECH. & POL'Y 347, 354.

33. *See In re* Bilski, 545 F.3d 943, 963 (Fed. Cir. 2008).

patents invalid.[34] Beyond this basic concern, the Federal Circuit has been faced with the task of applying a multitude of tests with little guidance as to the weight or relevance of each test. These tests arise from the principle that "laws of nature, natural phenomena, and abstract ideas," including mathematical algorithms, are unpatentable subject matter.[35] The tests applied by the courts in the years since *Bilski v. Kappos* include:

(1) **The Mental Steps Exception:** This test involves ascertaining whether an invention encompasses only methods that may be carried out in the human mind.[36] Such a method "is merely an abstract idea and is not patent-eligible under § 101."[37] The motivating sentiment behind this test is the unease felt toward a scheme that effectively makes a thought process legally protected.

(2) ***Flook*-like Inventive Concepts ("point of novelty"):** This analysis is similar to that done when looking for meaningful limitations to a possibly preemptory invention. There is, however, a subtle difference in formulation. Under this test, non-inventive concepts are first eliminated from the scope of the invention. The remaining components are assumed to be the invention for which protection is sought and they are what is analyzed under § 101.[38] This test has been chastised for bringing an inherent § 103 non-obviousness analysis[39] into the § 101 patentability analysis.[40]

(3) **The Machine-or-Transformation Test:** Although the Supreme Court relegated this test to a lower status, it remains a useful indicator of patentability.[41]

---

34. *See, e.g.*, Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2358 (2014) (stating that the "pre-emption concern . . . undergirds [the] § 101 jurisprudence"); Accenture Global Servs. v. Guidewire Software, Inc., 728 F.3d 1336, 1344–46 (Fed. Cir. 2013).

35. *See* Diamond v. Diehr, 450 U.S. 175, 185 (1981).

36. CyberSource Corp. v. Retail Decisions, Inc., 654 F.3d 1366, 1373 (Fed. Cir. 2011); *see also, e.g.*, PerkinElmer, Inc. v. Intema Ltd., 496 Fed. Appx. 65, 70–72 (Fed. Cir. 2012).

37. *Id.*

38. *See, e.g.*, Ariosa Diagnostics, Inc. v. Sequenom, Inc., 788 F.3d 1371, 1377 (Fed. Cir. 2015) (stating that for processes encompassing unpatentable natural phenomenon, "the [patentable] process steps are the additional features that must be new and useful").

39. Section 103 states that "[a] patent for a claimed invention may not be obtained . . . if the differences between the claimed invention and the prior art are such that the claimed invention as a whole would have been obvious before the effective filing date of the claimed invention to a person having ordinary skill in the art to which the claimed invention pertains." 35 U.S.C. § 103 (2012).

40. The Supreme Court acknowledged such criticism in *Flook*. *See* Parker v. Flook, 437 U.S. 584, 593 (1978).

41. *See, e.g.*, *PerkinElmer, Inc.*, 496 Fed. Appx. at 72–73; *CyberSource Corp.*, 654 F.3d at 1371. The machine-or-transformation test was expressed in *Benson* as the following: a "process patent must either be tied to a particular machine or apparatus or must operate to

(4) **Preemption and Meaningful Limitations:** This two-step test involves evaluating whether the patent too broadly preempts any further use of a natural law, natural phenomena, or abstract idea due to a lack of meaningful limitations.[42] First, a court determines whether the patent is drawn to one of the previously mentioned exceptions; second, a court asks if there are meaningful limitations that would prevent wholesale preemption.[43] Some preemption is a natural result of giving a monopoly on the practice of an invention. The purpose of this analysis, however, is to ensure that the preemption does not restrict future innovation too expansively.[44]

The primary problem is the vagueness regarding how patent eligibility under § 101 should be analyzed in light of these multiple tests. This vagueness leaves the Supreme Court with an incredible amount of flexibility to abruptly change course. The Federal Circuit is in the difficult position of stitching together vague and inconsistent precedent while knowing it can easily be rebuked. The USPTO has a tendency to err on the side of granting patents,[45] which means that this vagueness leads to a proliferation of low quality patents and produces a host of problems associated with software patents. These problems are described in Part II.B.

However, the USPTO may be able to utilize this same vagueness when analyzing software patents to screen out low quality patents. A necessary commonality between the tests is that they attempt to tease out whether a software invention is an unconstrained abstract idea and is thus unpatentable.[46] Another commonality is that they embody a preference for physicality.

The Supreme Court most recently addressed the issue of software patents in *Alice Corp. Pty. v. CLS Bank International*.[47] In *Alice*, the Supreme Court invalidated patent claims for a method of performing hedging under § 101 because the claims were "directed to an abstract idea"[48] and there was no inventive step transforming that idea into a

---

change articles or materials to a 'different state or thing'" to be patent eligible. *See* Gottschalk v. Benson, 409 U.S. 63, 71 (1972).

42. *See, e.g.*, Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2357–60 (2014).

43. *See id.*

44. *See id.*

45. *See, e.g.*, Jonathan Masur, *Patent Inflation*, 121 YALE L. J. 470, 474, 505 (2011).

46. Care must be taken in considering patentability and abstractness. A patentable invention may *contain* abstract ideas. *See, e.g.*, Diamond v. Diehr, 450 U.S. 175, 187 (1981); Mackay Radio & Telegraph Co. v. Radio Corp. of America, 306 U.S. 86, 94 (1939). Indeed, determining if a broad preemptory claim contains a "meaningful limitation" or some "inventive step" essentially involves analyzing whether an invention or claim containing an abstract idea is sufficiently constrained.

47. *Alice*, 134 S. Ct. at 2347.

48. *See id.* at 2356.

patent-eligible invention.[49] The Supreme Court held that "the mere recitation of a generic computer cannot transform a patent-ineligible abstract idea into a patent-eligible invention."[50] In rendering its decision in *Alice*, the Court refrained from positing a forward-looking standard. Instead, the Court determined that what was being patented was a "fundamental economic practice," analogized to the abstract concept that faced the court in *Bilski*, and held the patent claims in question to be abstract.[51]

Although the Supreme Court would not directly say so, the result in *Alice* may be interpreted as a resurgence of support for an altered machine-or-transformation test. The computer (or other electronic device capable of running software), however, is a machine and is the sole environment in which software operates. It is thus intrinsically tied to software, making futile any attempt to separate software from a computer for the purpose of patent analysis. Furthermore, the basic components of computers themselves have been sufficiently patented already. Therefore, the only remaining part to be patented in this machine-plus-software combination is the software. If only software remains to be patented, however, and the only machine that can operationalize software is a computer, then software itself is unable to meet the machine prong of the machine-or-transformation test. This is an example of the difficulties presented by the multitude of inconsistent and vague tests surrounding software patent eligibility. The existence of so many vague tests requires us to instead think normatively about software patent eligibility.

B. Software Patent Problems

The fundamental problem that policy makers and scholars express regarding software patents is that they hinder innovation,[52] in direct

---

49. *See id.* at 2357–58. Note that the Preemption and Meaningful Limitations test described above comes from the framework, established in *Alice*, of first finding whether the invention is drawn to an abstract idea, and second, finding whether it is limited enough to avoid preemption of the entire application of that idea.

50. *Id.* at 2358.

51. *See id.* at 2355–57; Annal D. Vyas, *Alice in Wonderland v. CLS Bank: The Supreme Court's Fantastic Adventure into Section 101 Abstract Idea Jurisprudence*, 9 AKRON INTELL. PROP. J. 1, 13–15 (2015). Numerous litigators, patent prosecutors, in-house counsel, and others embroiled in the issues surrounding software patents have echoed concerns regarding the lack of guidance provided by the Supreme Court's *Alice* decision. *See Where Do We Stand One Year After* Alice*?*, LAW360 (June 17, 2015 8:27 PM), http://www.law360.com/articles/668773/where-do-we-stand-one-year-after-alice [https://perma.cc/DB6X-2LAV].

52. *See, e.g.*, FED. TRADE COMM'N, TO PROMOTE INNOVATION: THE PROPER BALANCE OF COMPETITION AND PATENT LAW AND POLICY ch. 3 at 56 (2003), http://www.ftc.gov/os/2003/10/innovationrpt.pdf [https://perma.cc/3KF7-ALJ4]. Some simply find no evidence of a net benefit provided by software patents. *See* James Bessen, *A Generation of Software Patents*, 18 B.U. J. SCI. & TECH. L. 241, 261 (2012).

opposition to the Constitution's directive.[53] Some even argue that most software firms would be better off without patents.[54] A number of reasons have been proffered as to why patenting software is not ideal.[55] Below are four of the most prominent arguments.

1. The Patent System Does Not Suit Software

Software simply does not fit the patent system. The development and lifecycle of most software is short and technologies are quickly supplanted,[56] whereas patent prosecution is extremely slow, taking years.[57] Furthermore, the twenty-year term of most patents[58] is massively disproportionate to the lifespan of most software.[59] This works as a barrier to innovation by locking up ideas from those who could contribute to follow-on innovation.

---

53. *See* U.S. CONST. art. I, § 8, cl. 8 (stating that the purpose behind authorizing Congress to construct the patent system is "[t]o promote the Progress of Science and useful Arts").

54. *See, e.g.*, JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE 16 (2008) (concluding that after considering the effect of other patent owners and the risk of litigation, the average public firm "outside the chemical and pharmaceutical industries would be better off if patents did not exist").

55. For example, Google produces a new version of its Android mobile operating system and related Software Development Kit each year, sometimes twice in one calendar year. *See* Sarah Mitroff & Jessica Dolcourt, *The Android Era: From G1 to Lollipop*, CNET (May 8, 2014 1:00 PM), http://www.cnet.com/news/history-of-android [https://perma.cc/49TR-44CC].

56. *See, e.g.*, Eric Goldman, *The Problems with Software Patents (Part 1 of 3)*, FORBES (Nov. 28, 2012 2:53 PM), http://www.forbes.com/sites/ericgoldman/2012/11/28/the-problems-with-software-patents; U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 30 (stating that product development process at start-up company interviewed can be "as short as 2 months").

57. *See* U.S. PATENT & TRADEMARK OFFICE, PERFORMANCE AND ACCOUNTABILITY REPORT FOR FISCAL YEAR 2013, at 21 tbls. 2–3 (2013), http://www.uspto.gov/about/stratplan/ar/USPTOFY2013PAR.pdf [https://perma.cc/4NAR-AUCT] (showing that patent prosecution usually takes over two and a half years); Goldman, *supra* note 56 (stating that it takes four or more years to fully resolve a patent prosecution). The recent changes introduced by the America Invents Act exacerbated the problem by imposing measures that prolong the time it takes to acquire a patent, such as a provision allowing third parties to introduce prior art during examination and challenge a patent early on. *See* 35 U.S.C. §§ 122(e), 321–329 (2012). This has stretched an already unsuitably long process into a process that takes so long that Apple could develop the next two versions of its OS X operating system during its pendency.

58. *See* 35 U.S.C. § 154(a)(2) (2012).

59. It is informative to use the length that software enjoys support by its developer as a proxy for how quickly the software industry moves. Cisco's End-of-Life Policy shows that the scope of its support narrows each year following the end-of-sale date, with all support terminating after a scant five years. *See End-of-Life Policy*, CISCO http://www.cisco.com/c/en/us/products/eos-eol-policy.html [https://perma.cc/4WXP-EVYQ]. Microsoft does not have a strict lifecycle policy governing all of its products, but the end-of-support date for its operating systems generally hovers around ten years — half of a patent term. *See Windows Lifecycle Fact Sheet*, MICROSOFT, http://windows.microsoft.com/en-us/windows/lifecycle [https://perma.cc/2KMK-JMC3].

The software industry was "highly innovative and growing rapidly" prior to software patents becoming commonplace.[60] Scholars have found that the discontent of innovators themselves, particularly in the high-tech and software sector,[61] leads them to ignore patents when innovating and perform a scatter-shot of patent filings later.[62] Patent law is intended to foster innovation and scientific progress by providing inventors with a safe way to disclose their inventions, contribute to society's pool of knowledge, and thus spur further innovation.[63] Because innovators tend to ignore patents, they fail to make efficiency-maximizing decisions regarding research and development ("R&D") efforts and risk wasteful spending on recreating the wheel.[64]

Considering this poor fit, it should come as little surprise that most software firms do not patent at all.[65] James Bessen noted that from 1996 to 2006 the number of patent-holding startup firms — the same firms often thought of as the champions of innovation — declined.[66] This decline occurred at a time when it was well understood that software, including broad business methods such as the one in *State Street*,[67] is patentable. It likely shows that these innovators understand that patents and software often do not work well together or that patents are unnecessary for software innovation.

2. The USPTO Is Ill-Equipped

The examiners at the USPTO are not properly equipped to deal with the explosion in software patenting.[68] The USPTO itself is overworked and underfunded.[69] Software patent applications are among

---

60. James Bessen & Robert M. Hunt, *An Empirical Look at Software Patents*, 16 J. ECON. & MGMT. STRATEGY 157, 162 (2007).

61. *See* Bessen, *supra* note 52, at 242 (identifying a survey in which most software developers indicated opposition to software patents).

62. *See, e.g.*, U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 31; Mark A. Lemley, *Ignoring Patents*, 2008 MICH. ST. L. REV. 19, 21; TOM KRAZIT, *Why Tech Companies Want Engineers to Ignore Patents When Designing Products*, GIGAOM (Nov. 2, 2011, 6:00 AM), https://gigaom.com/2011/11/02/419-why-tech-companies-want-engineers-to-ignore-patents-when-designing-prod [https://perma.cc/6TZY-AKMK].

63. This fundamental concept of patent law is also a reason why novelty and non-obviousness are requirements for patentability. *See, e.g.*, DAN HUNTER, INTELLECTUAL PROPERTY 95, 98 (Dennis Patterson ed., 2012).

64. *See* Chien, *supra* note 2, at 294.

65. *See* Bessen, *supra* note 52, at 255.

66. *See id.* at 255 tbl.1.

67. *See supra* Part II.

68. *See* Bessen, *supra* note 52, at 253 fig.1 (showing that in 2009 alone about 40,000 software patents were issued — a large increase from the roughly 2000 issued in 1980). This should not be construed as a statement that the USPTO examiners are not well intentioned or hard working. It is simply a statement that examiners are overworked and given insufficient legal tools and institutional support.

69. *See, e.g.*, Joshua L. Sohn, *Can't the PTO Get a Little Respect?*, 26 BERKELEY TECH. L.J. 1603, 1623 (2011); Michael H. Davis, *Patent Politics*, 56 S.C. L. REV. 337, 370 n.131 (2004).

the most complex patents the USPTO has to examine.[70] Some have also claimed that the USPTO simply does not staff examiners with the knowledge required to parse software patents.[71] Beyond these institutional shortcomings, there is the more critical issue of insufficient legal tools.

The lack of legal guidance takes the form of a vague set of precedent given by the courts for interpreting a statutory regime that does not discuss software.[72] Although the USPTO has some autonomy in developing its own examination guidelines, such guidelines must fit within the legal framework. Without the requisite institutional and legal tools, it is virtually inevitable that large swaths of low quality software patents will find their way out the USPTO's door.

3. Software Patent Thickets

A patent thicket is "an overlapping set of patent rights requiring that those seeking to commercialize new technology obtain licenses from multiple patentees."[73] Thickets cause undue hold-up at firms, increase the cost of entry for new players, increase the cost of R&D, and suppress innovation.[74] While some have asserted that thickets are natural and not particularly concerning,[75] their arguments fail to recognize the software industry's uniqueness.

Software firms are sensitive to delays, and the first-mover market advantage gained by a firm is extremely important; it is often a sufficient incentive by itself to promote R&D.[76] Software patents generally have broad claims that are fuzzy and ambiguous, and thus do not give

---

70. Software patents often contain more claims than other patents. *See* Bessen & Hunt, *supra* note 60, at 170 tbl.2 (showing that software patents contain thirty-three percent more claims than other patents on average). Considering the incredible rate at which software patents have been filed, prior art searches have turned into an even more complicated and difficult process in the software realm. *See* FED. TRADE COMM'N, *supra* note 52, ch. 3, at 45–46 (noting that increased software patenting has "presented challenges in locating the relevant prior art").

71. Examiners also lack experience in and have trouble examining business methods, which are another vehicle through which numerous software patents arise. *See* Baird, *supra* note 32, at 355.

72. *See supra* Parts II & II.A.

73. Carl Shapiro, *Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard Setting*, 1 INNOVATION POLICY AND THE ECONOMY 119, 119 (Adam B. Jaffe et al. eds., 2001).

74. *See id.* at 124–26; Nieh, *supra* note 6, at 318–19; Shrestha, *supra* note 2, at 124–25.

75. *See* Sir Robin Jacob, Judge of the Court of Appeals of England and Wales, Patents and Pharmaceuticals — A Paper Given on 29th November at the Presentation of the Directorate-General of Competition's Preliminary Report of the Pharma-sector Inquiry (Nov. 29, 2008), http://ec.europa.eu/competition/sectors/pharmaceuticals/inquiry/jacob.pdf [https://perma.cc/R8A5-FMZP] (stating that thickets are "in the nature of the patent system itself" and that the thicket phenomenon "should happen and . . . has always happened").

76. *See* Robert E. Thomas, *Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law*, 25 SANTA CLARA COMPUTER & HIGH TECH. L.J. 191, 215 (2008).

clear notice of either the claimed invention or the patentee's rights.[77] Fuzzy and ambiguous claims increase costs imposed by the thicket because they increase the chance of a software developer paying the cost of either licensing or being litigated against for infringement of a patent that may not even cover the software created by the developer. The time sensitivity and patent-related transaction costs present in the software industry further deepen the effects of the patent thicket.

Many point out that the thicket problem "arises from the flood of patents that are granted by the USPTO each year."[78] Intuitively, having more patents results in a higher probability of overlaps — particularly when many innovators remain willfully ignorant of what is patented.[79] Considering that software patents are not generally required to incentivize software firms to innovate, the natural solution is for the USPTO to grant fewer software patents and diminish the growing thicket.

4. Software Patents Are Easily Abused

Several issues combine to make software patents relatively easy to abuse.[80] The primary reasons include the unclear and fuzzy boundaries of the property rights set by overly broad claims[81] and the ability of malicious litigators to take these patents and assert them systematically, efficiently, and cheaply.[82] Patent abuse is manifested in different ways. Abusers often instigate a barrage of litigation against numerous defendants — even those working on tangentially related subject matter.[83] Others essentially extort high licensing fees for bad patents on fundamental concepts that somehow passed muster during

77. *See id.* at 217–18; U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 28–30.

78. Shrestha, *supra* note 2, at 125.

79. *See supra* note 62.

80. I define the abuse of software patents as the assertion of a patent in an attempt to primarily restrict the R&D or innovative abilities of another party rather than for the purpose of protecting any property rights enshrined in the patent. This definition may be challenging to apply as it is based on the intent of the patent holder. However, there is little reason to lump innovators such as university research groups that may be more than happy to enter into mutually beneficial licensing agreements into the same group as those seeking to suppress competition or sit on low quality and overbroad patents in order to rent-seek.

81. *See* U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 28.

82. *See* Chien, *supra* note 2, at 292.

83. Patent-holding company Soverain used U.S. patents 5,715,314, 5,909,492, and sometimes 7,272,639, whose pertinent claims covered the idea of a "shopping cart," to sue a long list of online retailers. *See* Joe Mullin, *How Newegg Crushed the "Shopping Cart" Patent and Saved Online Retail*, ARS TECHNICA (Jan. 27, 2013, 4:00 PM), http://arstechnica.com/tech-policy/2013/01/how-newegg-crushed-the-shopping-cart-patent-and-saved-online-retail/ [https://perma.cc/Y8DK-JSXT]. After extracting $40 million from Amazon and an undisclosed amount from Gap, *see id.*, online retailer and defendant Newegg stood up and fought back. The Federal Circuit found that the claims encompassing the "shopping cart" idea in the '314 and '492 patents — in addition to other claims — were invalid and vacated the finding of infringement against Newegg. *See* Soverain Software LLC v. Newegg Inc., 705 F.3d 1333, 1341, 1344, 1346 (Fed. Cir. 2013).

the examination process at the USPTO.[84] Ultimately, the victim innovator must pour resources into licensing or litigating against bad patents rather than into socially beneficial innovation.

Such abuse is pronounced in the software industry because of the instability of the law. Parties are unsure as to the boundaries of the patentee's rights and what a court will construe falls within the claims of the abuser's patent.[85] This results from the insufficient legal framework given by Congress and the courts regarding software patents, as examined in Parts II and II.A.

In addition to being able to bundle their claims,[86] patent trolls have more flexibility in litigating because they need not fear retaliatory litigation. They are not focused on commercializing technology of their own.[87] These abusers often exert great pressure upon software innovators.[88] The combination of this pressure and the quick development cycles with resultant sensitivity to hold-ups often forces software developers to settle quickly to avoid costly litigation and delays.[89] A fundamental change is required to tackle the software patent problem.

---

84. An example is the litigation between Research in Motion (RIM) and non-producing patent-holding company NTP, Inc., in which RIM settled out of court by paying over $600 million to NTP for patents that were largely invalidated years later by the USPTO. *See* Rob Kelley, *BlackBerry Maker, NTP Ink $612 Million Settlement*, CNN MONEY (Mar. 3, 2006, 7:29 PM), http://money.cnn.com/2006/03/03/technology/rimm_ntp/ [https://perma.cc/YE6K-C974]; *Ex parte* NTP, Inc., No. 2008-004602, 2011 Pat. App. LEXIS 23859 (B.P.A.I. Dec. 21, 2011); *Ex parte* NTP, Inc., No. 2008-001116, 2011 Pat. App. LEXIS 23900 (B.P.A.I. Dec. 20, 2011); *Ex parte* NTP, Inc., No. 2008-004606, 2010 Pat. App. LEXIS 15609 (B.P.A.I. Feb. 4, 2010).

85. *See* U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 28–32 (mentioning that patentees of software patents often use overly broad functional claims that bring into the scope of the patent unimagined technologies not contemplated at the time of invention).

86. *See* Chien, *supra* note 2, at 292. "Between 2007 to 2011, 64 percent of defendants were sued over software-related patents, and these patents were at issue in the lawsuits that accounted for about 89 percent of the increase in defendants over this period." U.S. GOV'T ACCOUNTABILITY OFF., *supra* note 5, at 21. Non-practicing entities asserted software patents to a "much greater extent," and when grouping suits by defendant, "software-related patents were used to sue 93 percent of the defendants in PME suits." *Id.* at 21–22. This is a far cry from the 46 percent of defendants sued under software-related patents by operating companies. *See id.* at 22.

87. *See* Chien, *supra* note 2, at 292–93.

88. *See generally* Krish Gupta, *Patent Trolls Stifle Innovation and Business Investment*, EMC REFLECTIONS BLOG (June 26, 2014), http://reflectionsblog.emc.com/patent-trolls-stifle-innovation-business-investment/ [https://perma.cc/9Z38-WY93] (discussing the pressure patent trolls put on technology-producing corporations such as EMC).

89. *See generally* Lemley, *supra* note 62 (discussing hold-ups and sensitivity to "millions of dollars per case in legal fees"); Mullin, *supra* note 83 (explaining that "defendants tend to be driven to settle").

III. SOFT PHYSICALITY REQUIREMENTS AND SOFTWARE

One of the root causes of the software patent issue is the vague and generally uninformative legal framework given to the USPTO.[90] As an executive agency lacking substantive rulemaking authority, the USPTO is largely bound by the existing legal framework. The existing framework's vagueness, however, could allow the USPTO to be more proactive in denying software patents as impermissibly abstract. My proposal is to leverage this flexibility, as well as the Supreme Court's *Alice* decision, to introduce a requirement of soft physicality to software patent eligibility analysis. The proposal aims to alleviate the problems presented by software patents while allowing innovators that rely on patents, including software patents, to make use of the patent system.[91]

Some commentators have advocated for the return of physicality requirements in the claims or specifications of patents,[92] in line with now-outmoded tests.[93] Strict physicality requirements naturally provide much clearer guidance for USPTO examiners while working to prevent many impermissibly abstract software patents. This, however, may be too inexact a fit for the entirety of software-based inventions. Instead, I propose applying physicality standards that have been updated to be more appropriate in the context of software and current precedent.

Much of the discussion regarding the patent eligibility of software dances around the inherent abstractness of software.[94] The Merriam-Webster Dictionary defines "abstract" as something "relating to or involving general ideas or qualities rather than specific people, objects, or actions."[95] In the legal interpretation of abstractness applied to software, software patents are considered abstract. While some argue that software itself is not abstract,[96] that view is at odds with

90. *See supra* Part II.A.

91. Striking this balance is critical as research has indicated that, while there are numerous problems with software patents in the software industry, it is at least possible that software patents are beneficial to hardware innovators that obtain software patents. *See* Bessen, *supra* note 52, at 261.

92. *See, e.g.*, Thomas, *supra* note 76, at 238–41.

93. For example, the Federal Circuit's *In re Bilski* decision expounding the machine-or-transformation test as the sole test for the eligibility of a process under § 101 can effectively be read as requiring physicality: a tie into a physical machine or the physical effect of transforming an article. *See supra* Parts II & II.A.

94. *See, e.g.*, Bessen & Meurer, *supra* note 54, at 187; Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2360 (2014) (holding that the claims are abstract and thus cover non-statutory material although they include a computer system).

95. *Abstract*, MERRIAM-WEBSTER ONLINE DICTIONARY, http://www.merriam-webster.com/dictionary/abstract [https://perma.cc/RDS8-EQER].

96. *See, e.g.*, Seong-hee (Emily) Lee, *Software Patent Eligibility: A Call for Recognizing and Claiming Concrete Computer Programs*, 95 J. PAT. & TRADEMARK OFF. SOC'Y 402, 403–05 (2013).

scholars that maintain software is fundamentally abstract.[97] Indeed, it may be said that the actual written source code that embodies the idea protected by a software patent is tangible, that code falls within the domain of copyright law, not patent law. More fundamentally, however, the abstractness of software itself is not the issue — the issue is whether the claimed invention is abstract.

Software code is an abstraction of the physical working of the computer upon which the software runs; the processes of software generally do not relate to specific tangible objects or actions. However, software patents may be embodied as a process or a business method, and, because they represent software, these are patents that by necessity must be enabled through a physical apparatus, i.e., the computer. Nevertheless, the processes and methods underpinning software do not necessarily need to be claimed abstractly. It is possible to specify the software with varying levels of abstraction, with corresponding levels of specificity as to the mode of implementation. Thus, the abstractness of claims for software patents cannot be evaluated in isolation but rather requires reference to the specificity of its implementation. This effectively collapses the *Alice* two-step analysis into one step.[98]

Unfortunately, the courts have not provided a metric by which we can ascertain abstractness itself. Rather than giving a test, the courts generally follow a "we know it when we see it" totality-of-the-circumstances approach to finding abstractness.[99] It was against this uncertain backdrop that the USPTO and lower courts have had to analyze patents. Physicality requirements applied by the courts against software patents, such as those found in the machine-or-transformation test in *In re Bilski*, arose from simply applying requirements formulated and used in situations preceding the information technology revolution.[100]

---

97. *See* Bessen & Meurer, *supra* note 54, at 201–03.

98. This two-step analysis is used to determine if a patent claims patent-eligible applications of laws of nature, natural phenomena, and abstract ideas. First, the court finds whether the claim fits into one of those categories, and second, finds whether there is a limiting "inventive concept" in order to prevent preemption. This analysis given by the Supreme Court's decision in *Mayo* is summarized by the Court in *Alice*. *See Alice*, 134 S. Ct. at 2355. This is the Preemption and Meaningful Limitations test discussed in Part II.A, *supra*.

99. The courts often look to factors such as whether the patent encompasses "fundamental" ideas or commonly used principles and discusses preemptory concerns in finding that a patent is abstract. *See, e.g.*, *Alice*, 134 S. Ct. at 2356–57.

100. For example, the *Benson* court pointed to *Cochrane v. Deener*, 94 U.S. 780 (1876) (holding that patents covering a flour-sifting apparatus were valid), when laying the foundation for the transformation prong of the machine-or-transformation test. *See* Gottschalk v. Benson, 409 U.S. 63, 69–70 (1972). The foundation for the machine prong of the modern form of the test was laid by references in *Benson* to processes being "tied to a particular machine," *see id.*, at 70–71, and opinions such as *Smith v. Snow*, 294 U.S. 1 (1935) (holding that a patent for an improved apparatus and method for incubating eggs was valid).

Prior to the information technology revolution, there were fewer developments worth patenting within the gray area found between abstract inventions and concrete, useful inventions.[101] It was easier to declare a patent missing a physical component as not deserving of patent protection, as the risk of impeding innovation was low. This gray area has expanded with the development of the market for electronics and information technology, almost all of which is controlled in some regard by software, but the legal standards from a bygone era were simply transplanted without sufficient adaptation to the new context.[102]

With all the problems facing software patents, they should undoubtedly be required to meet higher standards than patents given in other fields[103] so that only those that are likely to promote innovation are permitted. The USPTO's adoption of translated physicality requirements to software will provide a much-needed hurdle to prevent innovation-sapping software patents from being granted. Although some have argued that *Alice* is too limiting on software patents — and it is true that many software patents have been invalidated in the wake of *Alice* — the Supreme Court's failure to provide a principled rule as to the patent eligibility of software is problematic.[104]

These translated soft physicality requirements, as I call them, mandate that to be statutorily eligible under § 101 and prevent a finding of impermissible abstractness, the claimed invention must contain a tie to the specific hardware or computing platform(s), whichever operates at the highest level yet still defines operationalization,[105] that

---

101. Although there were patents granted well before the software patent boom for things that could be described as having fuzzy boundaries, such as mechanical devices or chemicals, the nature of these inventions necessarily limited the scope of their protection by virtue of being represented in a definite physical form, chemical structure, or otherwise. *See* Bessen, *supra* note 4. Because software itself is conceptual and abstract, the patents covering software must allow for abstractness and must allow for more vagueness and fuzzier boundaries. *See id.*

102. *See, e.g.*, *Benson*, 409 U.S., at 69–71; Diamond v. Diehr, 450 U.S. 175, 181–84 (1981).

103. Pharmaceutical patents are often contrasted with software patents, with the pharmaceutical industry acting as an example of an industry that fits well with patent law. *See, e.g.*, Bessen, *supra* note 52, at 249. Patent law provides a large incentivizing push to innovate and develop new drugs and technologies in the pharmaceutical industry largely due to characteristics vastly different from the software industry, including the large amount of time and capital necessary to innovate in the pharmaceutical industry. One recent study found it takes $2.6 billion to develop and market a new drug. *See Cost to Develop and Win Marketing Approval for a New Drug Is $2.6 Billion*, TUFTS CENTER FOR THE STUDY OF DRUG DEVELOPMENT, (Nov. 18, 2014), http://csdd.tufts.edu/news/complete_story/pr_tufts_csdd_2014_cost_study [https://perma.cc/5GTT-PCZS].

104. *See infra* Part V.B for a more complete discussion.

105. "Highest level" refers to the level of distance away from the physical hardware of the device or system running the software in terms of a dependency chain. For example, a tie to a platform such as an operating system like Windows, OS X, or Linux is at a "higher level" than the hardware of the computer itself. Furthermore, a computing platform tie to a

the patentee intends the software to run upon. The innovation in this proposal is in the treatment of these platforms as equivalently limiting as physical hardware in the pre-information-technology world. This allows for the patenting of software in line with current standards.

The term "computing platform" has numerous definitions. For example, the Merriam-Webster Dictionary uses the rather narrow definition of "computer architecture and equipment using a particular operating system."[106] The meaning of computing platform I prefer to use, however, encompasses the hardware configuration, operating system, and other fundamental pieces of software that are required for the invention to operate.[107]

This proposal follows naturally from this definition of a computing platform. The platform or hardware upon which the software functions effectively gives the software functional or "practical" drive and is the vehicle through which the software is actualized. The requirement that patentees tie their inventions to the platform or hardware that enables practical use of the abstract software code gives three primary benefits.

First, it explicitly allows for software patents while preventing overly broad preemptive claims. Second, it allows others to contribute follow-on innovation without being blocked by preemptive software patents. Soft physicality requirements would drastically reduce the probability of a given patent preempting all possible uses for fundamental developments that could otherwise be utilized in further innovation. This is because requiring an explicit tie to current platforms necessarily prevents the patent from covering new and unanticipated technologies. Finally, and important for its own operationalization, explicitly formalizing soft physicality requirements will give the USPTO examiners the legal tools to quash innovation-stifling software patents before they enter the thicket.

IV. SOFT PHYSICALITY IN OPERATION

Following the Supreme Court's *Alice* decision, the Federal Circuit in *Content Extraction & Transmission LLC v. Wells Fargo Bank, National Ass'n* upheld a ruling that four patents which covered nothing more than the "abstract idea of extracting and storing data from hard

service such as the Apache Web Server is at a higher level than that of an operating system as such software runs on top of an operating system.

106. *Platform*, MERRIAM-WEBSTER DICTIONARY, http://www.merriam-webster.com/dictionary/platform [https://perma.cc/KLC9-Q25M].

107. *Cf. Platform*, FREE ON-LINE DICTIONARY OF COMPUTING (Dec. 7, 1994), http://foldoc.org/platform [https://perma.cc/EE6X-WZSK].

copy documents" are invalid as patent-ineligible under § 101.[108] This is representative of the kind of poor quality patents granted by the USPTO as a result of the inadequate legal tools at its disposal. Under the *Alice* framework, the Federal Circuit found the patents wanting for more concrete limitations, holding that "the mere recitation of a generic computer" is insufficient.[109]

Similarly, in *buySAFE, Inc. v. Google, Inc.* the Federal Circuit affirmed a ruling that a patent covering the concept of "a third party guarantee of a sales transaction" and simply applying it "using conventional computer technology and the Internet"[110] is invalid under § 101.[111] The Federal Circuit, again relying on *Alice*, noted the insufficiency of "merely requir[ing] generic computer implementation."[112] The Court went on to explain that the usage of a generic computer adds no inventive concept to the abstract idea, and does not sufficient-

---

108. Content Extraction & Transmission LLC v. Wells Fargo Bank, Nat'l Ass'n, 776 F.3d 1343, 1349 (Fed. Cir. 2014). The representative claim of the patents in question is as follows:

> A method of processing information from a diversity of types of hard copy documents, said method comprising the steps of:
> receiving output representing a diversity of types of hard copy documents from an automated digitizing unit and storing information from said diversity of types of hard copy documents into a memory, said information not fixed from one document to the next, said receiving step not preceded by scanning, via said automated digitizing unit, of a separate document containing format requirements;
> recognizing portions of said hard copy documents corresponding to a first data field; and
> storing information from said portions of said hard copy documents corresponding to said first data field into memory locations for said first data field.

Id. at 1345.

109. *Id.* at 1348 (quoting Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2358 (2014)).

110. buySAFE, Inc. v. Google, Inc., 765 F.3d 1350, 1352 (Fed. Cir. 2014) (quoting buySAFE, Inc. v. Google, Inc., 964 F. Supp. 2d 331, 335–36 (D. Del. 2013)).

111. *Id.* at 1355. The representative claim of the patent in question is as follows:

1. A method, comprising:

> receiving, by at least one computer application program running on a computer of a safe transaction service provider, a request from a first party for obtaining a transaction performance guaranty service with respect to an online commercial transaction following closing of the online commercial transaction;
> processing, by at least one computer application program running on the safe transaction service provider computer, the request by underwriting the first party in order to provide the transaction performance guaranty service to the first party,
> wherein the computer of the safe transaction service provider offers, via a computer network, the transaction performance guaranty service that binds a transaction performance guaranty to the online commercial transaction involving the first party to guarantee the performance of the first party following closing of the online commercial transaction.

Id. at 1351–52.

112. *Id.* at 1354 (quoting *Alice*, 134 S. Ct. at 2357).

ly limit the concept in the patent such that it may receive patent protection.[113]

With the framework I propose, the USPTO would have been equipped with the proper legal tools not only to strike these patents down during examination, but also to give clearer guidance to applicants regarding the rejections. The patent in question in *Content Extraction* fails to meet soft physicality requirements because references to such things as "digitizing units" and "memory" are extremely vague.[114] Indeed, it also used typical circular descriptions for an "application" comprising a "processor" that processes information and a "formatter" for "formatting."[115] Such claims would fail the test for lacking specifics on the hardware and platform required for operation. The level of specification should be such that it provides an ordinary artisan the ability to understand the scope of the invention and how it is actualized. The claims in the patent in question in *Content Extraction* only give a generic theoretical operation, leaving an incredibly broad range of possibilities for actualization; they do not allow the ordinary artisan to glean much useful information regarding scope.

This proposal also balances the hardware manufacturers' need to patent inventions that encompass embedded software.[116] Hardware manufacturers are a group of innovators that are generally thought of as being able to appropriately leverage patent law to further innovation.[117] By requiring patentees to specify the required hardware or computing platform(s), hardware manufacturers may still obtain patents by including the platform the software was designed to operate upon.

---

113. *See id.* at 1354–55.

114. *See* U.S. Patent No. 5,258,855, claim 70 (filed Mar. 20, 1991). A digitizing unit is a term for any device that takes as input an analog signal, which is a signal that exists on a continuous spectrum of voltages, and transforms it into a digital signal that ideally only contains two voltage levels representing binary 0 and 1. Thus, the term "digitizing unit" is a gloss that covers the behavior of numerous physical devices without giving any information as to implementation, operable voltage levels, efficiency, or any other relevant design features that may distinguish one digitizing unit from another digitizing unit.

115. *See id.*

116. Software that is embedded into a hardware device such as a television, optical disc drive, television remote control, etc., in persistent memory which performs functions that are generally much more limited in scope than the general pool of software available for and used on a general-purpose computer is referred to as "firmware." *See Firmware*, TECHOPEDIA, http://www.techopedia.com/definition/2137/firmware [https://perma.cc/52EK-F74S]. Firmware is generally very application-specific and thus has a closer tie to the hardware *platform* into which it is embedded, easily satisfying soft physicality requirements.

117. *See, e.g.*, Bessen, *supra* note 52, at 261 (stating that although software patenting is concerning, "software patents might be highly beneficial to the various hardware industries that obtain large numbers of software patents"). Due to the increased time and capital investment typically involved in the research, development, design, and manufacturing of hardware, there is likely to be a larger incentivizing role for patent law to approve such investments.

A. Precedent and Soft Physicality

Soft physicality fits within the ambiguities of the legal framework given by the Federal Circuit and the Supreme Court. As an executive agency, the fundamental job of the USPTO is to execute the law as laid out by the legislature and interpreted by the judiciary.[118] However, the USPTO lacks the broad substantive rulemaking authority that many other agencies, such as the Securities and Exchange Commission, enjoy.[119] More precisely, the USPTO lacks "the capacity to issue binding substantive rules,"[120] in large part because the USPTO generally does not receive the weighty *Chevron* deference to its rulemaking.[121] Nevertheless, when the law is vague, as is the case here, the USPTO has some flexibility in interpreting the bounds of the ambiguity and may even be able to rely on the deference it does receive from courts to shape substantive patent law.[122]

In *Alice*, the Supreme Court made it clear the primary motivating sentiment underlying the exclusion of abstract ideas is "one of preemption."[123] The Court went on to find that the abstract invention in *Alice* was not sufficiently limited by the use of a general-purpose computer[124] — in other words, a general-purpose computer did not constitute an appropriate "inventive concept" that would have rendered the patent eligible under § 101.[125] Nearly all software, with the exception of software intrinsically tied to the hardware upon which it is affixed (i.e., firmware), runs upon general-purpose computers or similar. To say that it is insufficient to take software, itself abstract,[126] and apply it to a computer implies a requirement to tie the patent for such software, or the process embodied by the software, to the animating framework which the software runs upon.

In *Bilski*, the Supreme Court relied on *Diehr* but instructed, however, that patentees may not circumvent the prohibition against patenting abstract ideas by the simple addition of "insignificant postsolution activity" or by "limiting the use of an abstract idea 'to a particular

---

118. While delegated legislative authority is another angle through which executive agencies may operate slightly outside the legal framework given by the judiciary and legislature, the USPTO lacks substantive rulemaking authority, which makes such an argument a non-starter. *See* John M. Golden, *The USPTO's Soft Power: Who Needs* Chevron *Deference?*, 66 SMU L. REV. 541, 542 (2013).

119. *See, e.g.*, *id.*

120. *Id.* at 545.

121. *See id.* at 550–51.

122. *See id.* at 553–58.

123. Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2354 (2014).

124. The patent encompassed the idea of reducing settlement risk through intermediated settlement. *Id.* at 2356.

125. *Id.* at 2357.

126. See *supra* Part III for the discussion of the abstract nature of software.

technological environment.'"[127] The addition of "insignificant postsolution activity" suggests an analysis under the non-obviousness and inventive step requirement of § 103.[128] Indeed, the Supreme Court's *Alice* decision fails to elaborate on this point. Furthermore, this was made as an offhand remark in the dicta of the *Diehr* opinion, from which the *Bilski* Court cited.[129] The requirement of including the hardware or platform(s) relied upon by the invention dovetails with this prohibition and acts as a litmus test to show whether the patentee truly has only trivial "insignificant postsolution activity."

The Supreme Court's unfavorable view of limiting a patent through ties to a "particular technological environment"[130] is rooted in a concern about preemption. In *Alice*, the Supreme Court explicitly noted that this and the previously discussed concern with postsolution activity "accords with the pre-emption concern that undergirds [the] §101 jurisprudence."[131] The best reading of the Supreme Court's dismissive reaction in *Alice* to the patentee's attempt to gain approval for its abstract patent by implementing it on a computer is that recitation of a general-purpose computer as a limiting environment is insufficient. It should not be read to mean that any recitation of specific technological environments is categorically insufficient.

Indeed, this must be true when embedding patentable processes and business methods in software because these ideas are naturally tied to the technological environment of software. The Supreme Court explicitly stated that an application of an abstract idea may receive patent protection if the claims "supply a 'new and useful' application of the idea."[132] The "'new and useful' application of the idea" is precisely how the software is implemented: it specifies what hardware and computing platforms the software runs on and in what manner the hardware and computing platforms interact with the software.

The question then is how narrow the limitation must be. Put another way: how limiting must the "environment" be? Requiring a patentee to specify in the claims the envisioned hardware or platform(s) used to operationalize the invention, and their interactions, furnishes a sufficient limitation on the patent while still allowing for appropriate software patents. Whether the usage of the hardware and platform(s) is novel or is something well-understood and routine would then be properly asked under the § 103 non-obviousness analysis.[133]

---

127. Bilski v. Kappos, 561 U.S. 593, 610–11 (2010) (quoting Diamond v. Diehr, 450 U.S. 175, 191–92 (1981)); *Alice*, 134 S. Ct. at 2358 (quoting *Bilski*, 561 U.S. at 610–11).

128. For the text of § 103, refer to *supra* note 39.

129. *See Diehr*, 450 U.S. at 191–92.

130. *See id.*

131. *Alice*, 134 S. Ct. at 2358.

132. *Id.* at 2357 (quoting Gottschalk v. Benson, 409 U.S. 63, 67 (1972)).

133. *See* 35 U.S.C. § 103, *supra* note 39.

The theoretical underpinning of the Supreme Court's current framework in *Alice* supports soft physicality. The analysis of patent eligibility under § 101 for software patents can be collapsed into a single test due to software's inherent abstractness and because the test naturally checks for such abstractness. Furthermore, soft physicality requirements work to address the fundamental issue of preemption and ensure that only appropriate software patents are given patent protection.

The Japanese patent system, discussed in the following Part, slowly expanded to cover software-related inventions like its American counterpart. In Japan, a series of examination guidelines equip the Japan Patent Office ("JPO") to efficiently examine software patents. Although the USPTO produces guidelines, they are not on the same level of specificity and utility.[134] The USPTO should similarly take a stronger stance when interpreting the legal framework to bolster its examiners' legal toolkit.

B. Tailored Standards in Other Jurisdictions

The purpose of the Japanese patent system is to leverage innovation to contribute to the development of industry.[135] Under Japanese patent law, an invention must be a "highly advanced creation of technical ideas" that "utilize[s] the laws of nature"[136] to be statutorily eligible for patent protection.[137] Further, an invention must also be "industrially applicable."[138] Software patenting may seem difficult to achieve given these requirements.[139] Japan, however, has followed the Western trend of increasingly providing protection for software, albeit at a slower pace.[140]

Most of the rules regarding how software patents are examined in Japan have been laid down by a series of Examination Guidelines.

---

134. The USPTO examination guidelines after the *Alice* decision simply recite the salient information from that decision and analogous details from previous decisions. *See* USPTO, 2014 INTERIM GUIDANCE ON PATENT SUBJECT MATTER ELIGIBILITY (Dec. 2014), http://www.gpo.gov/fdsys/pkg/FR-2014-12-16/pdf/2014-29414.pdf [https://perma.cc/XMC5-X4JB]. The outgoing JPO examination guidelines, however, offered standards that were useful when applied to unforeseen technologies or during claim construction. *See* JPO EXAMINATION GUIDELINES, *supra* note 8, at 10–13.

135. *See* Tokkyohō [Patent Act], Law No. 121 of 1959, art. 1 (Japan).

136. *See id.* art. 2, para. 1.

137. Scholars have taken this requirement to mean that an invention must use the laws of cause and effect that lie behind natural phenomena. *See* NOBUHIRO NAKAYAMA, TOKKYOHŌ 98 (2nd ed. 2012).

138. *See* Tokkyohō, *supra* note 135, art. 29, para. 1.

139. In fact, one of the preeminent scholars of Japanese patent law, Nobuhiro Nakayama, has said that computer software may be the area that presents the most problems in determining whether or not developments "utilize the laws of nature." *See* NAKAYAMA, *supra* note 137, at 103.

140. *See* ROBERT P. MERGES & JOHN F. DUFFY, PATENT LAW AND POLICY: CASES AND MATERIALS 199 (6th ed. 2013).

The first Guidelines arrived in 1975 and noted that software is "exceedingly abstract" and outside the bounds of protection.[141] Ultimately, software patents were allowed if the software was embedded as part of a patent-eligible method or product.[142] The acceptance of granting patents over software gradually increased,[143] and two guidelines published in 1997 and 2000 established the modern approach.

The revised 2000 Examination Guidelines for Patent and Utility Model is the current version. This revision of the Guidelines effectively marked the point at which software is rendered eligible for patent protection.[144] An amendment of the Patent Act in 2002 codified the stance taken by the 2000 Guidelines and extended protection to digitally distributed software.[145] The standard used for examinations conducted until October 1, 2015 states that a software-related invention is covered by statute if "information processing by [the] software is concretely realized by using hardware resources."[146] Under this tailored standard it becomes necessary to specify the "inputs, outputs, hardware resources as actors such as CPU, memory, computer and server and how to process the inputs by the hardware resources to generate the outputs for all elements" in a claim.[147] This requirement, tailored by the JPO for use when examining software patents, is similar to, yet weaker than, the proposal of soft physicality requirements.

With the clarifications the Guidelines offer,[148] examiners are better able to strike down ineligible patents.[149] Furthermore, the Japanese

---

141. JAPAN PATENT OFFICE, KONPYŪTA PUROGURAMU NI KANSURU HATSUMEI NITSUITE NO SHINSAKIJUN (SONO1) [Examination Guidelines for Inventions Related to Computer Programs, Part 1] 22 (Dec. 1975), http://www.furutani.co.jp/office/ronbun/soft-standard-1.pdf [https://perma.cc/5J8W-JD3F] (describing software as "極めて抽象的").

142. *See* NAKAYAMA, *supra* note 137, at 104. Allowing a patent in this way ensured that even inventions containing software — an abstract component — would adhere to the requirement that a law of nature be utilized.

143. In the decades following the 1975 Examination Guidelines, two subsequent guidelines were issued, each liberalizing the patenting of software. The December 1982 Application Guidelines on Inventions Related to Microcomputer Technology and the Treatment of Examination of Operating System-Related Technologies gave inventions related to microcomputers patent protection. *See id.* The July 1993 Examination Guidelines for Patent and Utility Model stated that, either when data processing using a natural law is performed by software or when hardware resources are used, the invention would be eligible for protection. *See id.* The 1993 Guidelines effectively allowed for software such as word processors to become patent-eligible. *See* James S. Sfekas, *Controlling Business Method Patents: How the Japanese Standard for Patenting Software Could Bring Reasonable Limitations to Business Method Patents in the United States*, 16 PAC. RIM L. & POL'Y J. 197, 206 (2007).

144. *See* Kazuyuki Motohashi, *Software Patent and Its Impact on Software Innovation in Japan*, RIETI Discussion Paper Series 09-E-038 4 (2009), http://www.rieti.go.jp/jp/publications/dp/09e038.pdf [https://perma.cc/8BUU-JTQK].

145. *See id.*; Tokkyohō, *supra* note 135, art. 2, para. 3; NAKAYAMA, *supra* note 137, at 105.

146. JPO EXAMINATION GUIDELINES, *supra* note 8, at 10.

147. ZENTARO KITAGAWA, 3-6 DOING BUSINESS IN JAPAN § 6.02(6)(a)(v)(A) (2016).

148. The outgoing examination guidelines on software-related inventions is seventy-seven pages and contains numerous examples of when software patents should and should not be granted. *See generally* JPO EXAMINATION GUIDELINES, *supra* note 8.

system shows that software-related patents can be treated differently from other patents in an innovative country with a modern legal system. The stricter requirements for software patents have not brought about a precipitous fall in the patenting of all software-related inventions. In fact, Japan has issued increasing numbers of patents that encompass software inventions.[150] What the system does provide, however, is a clear set of tools for examiners to use that are tied to standards tailor-made to fit an industry in which patents are an uneasy fit and often do not provide much additional incentive to innovate.[151]

Tailored standards allow for patents on appropriate inventions while also giving examiners the ability to reject patents that are unlikely to promote innovation. The USPTO should implement (as Japan has) a higher bar for software-related patents to ensure that what is patented is worth "the embarrassment of an exclusive patent."[152] As discussed in Part II.A, there is much room for case law interpretation to achieve this end due to the multitude of patent-eligibility tests, which are particularly vague and inconsistent when applied to software patents. The USPTO should leverage this vagueness and, to prevent a disconnect between the USPTO's practice and the expectations of inventors, issue policy guidelines indicating that they will take a more stringent stance on software patents in line with a requirement of soft physicality.

V. ARGUMENTS AGAINST SOFT PHYSICALITY REQUIREMENTS

Two fundamental criticisms that may be levied against the proposal of soft physicality are briefly discussed below.

---

149. *See, e.g.*, *id.* at 42–50 (showing examples of hypothetical inventions and how the Guidelines regarding software-related inventions should be applied in rejecting or accepting certain claims).

150. Most software patents are contained within the G06 subdivision of the International Patent Classification. While not necessarily encompassing only software-related patents, the number of patents awarded under this category roughly correlates with the number of software-related patents. In 2013 alone the JPO awarded over 18,000 patents under the G06 subdivision, up from around 5000 per year between the years 2000 to 2005. *See* JAPAN PATENT OFFICE, TOKKYO GYŌSEI NENJIHŌKOKUSHO 2014 NENBAN (TŌKEI・SHIRYŌHEN) [2014 Annual Administrative Patent Report (Data & Statistics Section)] 21 (2014), http://www.jpo.go.jp/shiryou/toushin/nenji/nenpou2014/toukei/dai-2.pdf [https://perma.cc/KF6G-DLZQ]; Japan Patent Office, TOKKYO GYŌSEI NENJIHŌKOKUSHO 2004 NENBAN (TŌKEI・SHIRYŌHEN) [2004 Annual Administrative Patent Report (Data & Statistics Section)] 3 (2004), http://www.jpo.go.jp/shiryou/toushin/nenji/nenpou2004_pdf/toukei/02-04-02.pdf [https://perma.cc/BJL7-Y7W3]. This uptick can be attributed to the more liberal stance taken toward patenting software that picked up steam in the early 2000s via the 2000 Examination Guidelines revision and the 2002 amendment to the Patent Act.

151. *Cf.* Motohashi, *supra* note 144, at 13.

152. Graham v. John Deere Co. of Kansas City, 383 U.S. 1, 10–11 (1966) (quoting Thomas Jefferson, Letter to Isaac McPherson (Aug. 13, 1813), *in* 13 THE WRITINGS OF THOMAS JEFFERSON 326, 334–35 (Andrew A. Lipscomb ed., 1903)).

A. Not All Software Is Abstract

Some argue that software is not inherently abstract.[153] It would be difficult, however, to find someone who would argue that a novel's story is not abstract. Conversely, the pages, ink, and exact arrangement of words within the book are a physicality implementing the story. The processes and methods embodied in a software patent are equivalent to the story of a novel; the software code and computer hardware is the paper, ink, and expression. Those arguing software is not abstract effectively argue that a story is self-actualizing.

These commentators claim that software is not abstract, despite the fact that it is essentially a set of instructions, because in reality software is confined by various limitations, such as hardware constraints, compatibility requirements, and human-introduced restrictions.[154] This, however, ignores the reality that a computer system may be precisely reduced in a simulator because these digital systems are well defined.[155] Thus, digital systems should not be thought of as intractable in the same sense that we might think of applications of the sciences as intractable. While in practice software might not operate exactly as the developer envisioned, the fact remains that the developer, if given enough time, could fully predict the operation of software prior to runtime. By contrast, the application of hard sciences is intractable because we simply do not have a perfect understanding of the natural world due to limitations in both human knowledge and measurement. It is conceivable that in the future we might have perfect conceptual knowledge of all scientific principles employed in the applied sciences like engineering, but, unlike the operation of software, we currently do not.

Abstract ideas must be actualized; that is the nature of abstractness. Soft physicality requirements ensure that such abstract ideas are pulled closer to actualization before they are patentable. Soft physicality leaves patent claims that are sufficiently actualized untouched by

---

153. *See, e.g.*, Lee, *supra* note 96, at 403–05 (arguing that software is not abstract because, unlike a mathematical formula, its behavior becomes increasingly unpredictable as it grows in complexity).

154. *See, e.g.*, *id.*

155. For example, the MIPS processor architecture is a popular architecture to use for teaching computer design, digital systems, and assembly language (the lowest level programming language besides coding directly in binary). As a result there are numerous simulators of MIPS processors such as SPIM, *see* James, Larus, SPIM: A MIPS32 SIMULATOR, http://spimsimulator.sourceforge.net/ [https://perma.cc/8Y89-G7N6], and MARS, *see MARS (MIPS Assembler and Runtime Simulator)*, MO. ST. U., http://courses.missouristate.edu/kenvollmar/mars/index.htm [https://perma.cc/MG5V-5XRW] (last modified Oct. 28, 2014). Other architectures and digital systems are similarly reducible, but there is generally not such a great commercial need for simulation and emulation and the speed of simulated architectures generally pales in comparison to the native hardware performance.

referencing appropriately limiting hardware or platforms.[156] Thus, soft physicality operates only against bare abstract patents that seek to broadly cover the story of an invention.

B. The Alice *Framework Is Sufficiently Limiting*

Some pro-patent commentators have denounced *Alice* as having overstepped in restricting legitimate software patents.[157] Others may decide that the *Alice* framework sufficiently limits software patents. Those commentators would likely suggest that soft physicality requirements are unnecessary. It is true that *Alice* has been a critical element contributing to the current wave of invalidations of software patents.[158] From a consequentialist point of view, this what we should primarily be concerned about. However, the consequentialist overlooks the utility of having clear positive standards.

The Supreme Court in *Alice* reached the proper decision, but the explanation given was insufficient.[159] Ultimately, the Court in *Alice* failed to provide a principled reason for *why* and *when* software patents are invalid. The Court gave no forward-thinking guidance and simply stated that recitation of generic components of a general-purpose computer used trivially is not sufficient to make an abstract idea patentable.[160] While this provides another stepping stone to support the invalidation of improvident software patents, such guidance framed in the negative only provides a sole data point rather than a generally applicable rule or framework.

What the USPTO needs, and what lower courts will benefit from, are positive rules. Soft physicality provides a positive rule, comports with the Supreme Court's preemption anxiety and latent fixation on physicality, and will assist the USPTO's examiners in ascertaining whether a given software patent is likely to foster or suppress innovation. Not only will such a positive rule provide a more principled means by which software patents may be challenged, such a rule will

---

156. Again, "appropriately limiting" means that an ordinary artisan would be able to glean from the claims the intended modes of implementation and thus the intended scope of the patent.

157. *See, e.g.*, Gene Quinn, *A Software Patent Setback:* Alice v. CLS Bank, IPWATCHDOG.COM (Jan. 9, 2015), http://www.ipwatchdog.com/2015/01/09/a-software-patent-setback-alice-v-cls-bank/id=53460/ [https://perma.cc/S68E-JKDQ].

158. *See* Jasper L. Tran, *Software Patents: A One-Year Review of* Alice v. CLS Bank, 97 J. PAT. & TRADEMARK OFF. SOC'Y 532, 539–41 (2015).

159. Others have likewise noted that the *Alice* decision provided very little in the way of positive tests for determining the abstractness of an invention. *See* Erika Harmon Arner, Elliot C. Cook & C. Gregory Gramenopoulos, *IP Update:* Alice Corp. v. CLS Bank*: The Supreme Court Weighs In on Patent–Eligibility*, FINNEGAN (June 20, 2014), http://www.finnegan.com/IPUpdateAliceCorpvCLSBankTheSupremeCourtWeighsInOn PatentEligibility/ [https://perma.cc/DJ8J-UKP4] (noting the court "expressly refrained from providing a test for determining whether an invention is directed to an 'abstract idea'").

160. *See supra* Part II.A, specifically note 51.

also create more certainty for patent holders and innovators seeking to patent inventions encompassing software while simultaneously increasing the quality of granted software patents.

The ambiguity surrounding the patent eligibility of software may also be viewed by some as a variant of beneficial or necessary flexibility. Although software patents may require flexible standards that can keep pace with the high-speed technological progress that software undergoes, the utility of that approach depends ultimately on the parties implementing the standards. As some scholars have pointed out, the USPTO generally errs on the side of improperly granting patents.[161] Thus, the USPTO will use flexible standards in a way that increases improperly granted patents. The USPTO's proactive adoption of a robust and forward-thinking positive standard — the adoption of soft physicality requirements — will give the USPTO a tool to determine the patentability of software in a more predictable and principled manner. Furthermore, a clearer standard will cut off the risk of granting improper patents due to the agency's natural inclination to stretch flexible standards towards being too permissive of low quality software patents.

VI. CONCLUSION

It is no secret that software patents present unique problems given their abstractness and ease of abuse. The USPTO and various courts have played tug of war over the formulation of standards, leaving patent examiners and lower courts with an inconsistent, unclear, and fundamentally deficient legal toolbox. Although the courts have thus far been reluctant to explicitly state that software is a unique industry requiring custom-tailored patent standards, this is precisely what must be done. The USPTO has the opportunity to take the first step by adopting soft physicality requirements during examination. While traditional physicality requirements are indeed improper for software, the tailored requirement of soft physicality will help correct the root problems of software patents and ensure that only software patents that truly "promote the Progress of Science and useful Arts" are granted.[162]

161. *See* Masur, *supra* note 45, at 474.
162. U.S. CONST. art. I, § 8, cl. 8.