From State Street Bank to CLS Bank and Back: Reforming Software Patents to Promote Innovation

ABSTRACT

For the past several decades, the Supreme Court and Federal Circuit have struggled to determine if, and under what circumstances, software is patentable. Once again, the Federal Circuit had an opportunity to provide clarity when it granted en banc review in CLS Bank. The resulting opinion contained a cursory per curiam decision and numerous concurrences and dissents, showing that the question is far from answered. Ultimately, the struggle over software patentability is not itself the problem, but a symptom of other problems in the patent system. Specifically, other substantive requirements of patentability are not weeding out overly broad patents because the person having ordinary skill in the art (PHOSITA) of software is assumed to be extremely skilled. Where PHOSITAs are highly skilled, the US Patent and Trademark Office (USPTO) allows broad claims based on minimal disclosure. By repairing the conception of the PHOSITA in the courts, the USPTO can issue narrower, less-restrictive patents. Once claim scope is adequately limited and more thorough disclosure required, the courts can return the patentability analysis to the permissive one in State Street Bank, ensuring public access to the invention and removing the cloud of suspicion that currently hangs over software patents.

TABLE OF CONTENTS

I. PATENT DOCTRINES AND SOFTWARE CHALLENGES......... 429
   A. First Principles of Patent Law .................................. 429
   B. The Problem of Software ......................................... 432
      1. Reverse Engineering ........................................... 433
      2. The Software PHOSITA .......................................... 436
      3. Examination Issues ............................................. 439
   C. The Genesis of Patentable Subject Matter Doctrine ...... 441

II. CLS BANK V. ALICE CORP. ........................................ 445
    A. Lourie’s Concurrence—Coextensivity .......................... 446
    B. Rader’s Dissent .................................................. 448
III. NEEDED REFORMS AT THE USPTO AND IN THE COURTS..... 449
A. A Return to State Street Bank........................................ 450
B. Reconsidering the Order of Validity Challenges .......... 452
C. Deference to Trial Court PHOSITA Determinations .... 453
D. Reforms at the USPTO.................................................. 454

IV. CONCLUSION .................................................................... 455

“What’s the shortest way to travel from Rotterdam to Groningen?”

That simple question, posed to the young Dutch computer scientist Edsger Dijkstra, sparked the invention of a simple algorithm to find the shortest path between two points on a map. Dijkstra invented the algorithm that now bears his name in 1956 to demonstrate the capabilities of the then-nascent field of digital computing. Today, Dijkstra’s algorithm provides the foundation of software that generates driving directions and routes information around the Internet. Thousands of algorithms like it surround users every day, returning Internet search results, recommending purchases, and selecting songs based on personal music preferences. Apart from Internet technologies, these algorithms time traffic lights, automate stock trading, and route phone calls. All of these technologies embody innovative algorithms and apply them in our everyday lives.

Curiously, despite the omnipresence of algorithm-driven technology, it is unclear whether, or in what circumstances, US law

2. See id.
permits patents on those algorithms.\textsuperscript{7} Specifically, controversy surrounds whether algorithms fall within the allowable categories of patentable subject matter.\textsuperscript{8} Congress intended the patent laws to permit patents on “anything under the sun that is made by man.”\textsuperscript{9} Despite this seemingly unlimited scope, courts have subsequently excluded from the universe of inventions those that are merely “laws of nature, physical phenomena, and abstract ideas.”\textsuperscript{10} Over the past forty years, the Supreme Court and the US Court of Appeals for the Federal Circuit have struggled to determine whether software algorithms are “abstract ideas.”\textsuperscript{11} Even the most recent Supreme Court decision to address the issue merely rejected the Federal Circuit’s proposed rule and failed to provide any guidance on what an appropriate rule should be.\textsuperscript{12}

A related issue arises when the US Patent and Trademark Office (USPTO) does issue software patents.\textsuperscript{13} Those patents contain broad claims, yet disclose little about how the patented technology works.\textsuperscript{14} Thus, patentees often gain powerful patent monopolies while providing the public with little useful knowledge about how those patented inventions work.\textsuperscript{15} As a result, the federal government is issuing patents, but the “general store of knowledge” is not growing.\textsuperscript{16}

\begin{footnotes}{
7. See, e.g., Mark A. Lemley et al., Life after Bilski, 63 STAN. L. REV. 1315, 1317–18 (2011) (“The patentability of software . . . has a long and tortured history.”).
8. See, e.g., Bilski v. Kappos, 130 S. Ct. 3218, 3229–30 (2010) (holding that a software system for hedging risk of commodity prices was not patent eligible); Bancorp Servs. L.L.C. v. Sun Life Assurance Co. of Canada (U.S.), 687 F.3d 1266, 1280–81 (Fed. Cir. 2012) (holding that a software system for managing life insurance policies is not patent eligible); CLS Bank Int’l v. Alice Corp. Pty. (CLS Bank II), 685 F.3d 1341, 1356 (Fed. Cir.) (holding that a software system for settlement of financial obligations was not patent eligible), \textit{vacated}, 484 Fed. Appx. 559 (Fed. Cir. 2012); cf., e.g., Ultramercial, LLC v. Hulu, LLC, 657 F.3d 1232, 1330 (Fed. Cir. 2011) (holding that a software system for delivering advertising to users is patent eligible), \textit{vacated sub nom.} Wildtangent, Inc. v. Ultramercial, LLC, 132 S. Ct. 2431 (2012) (mem.).
11. See, e.g., Bilski, 130 S. Ct. at 3227.
12. See id.
14. See infra Part I.B.
15. See infra Part I.A.
16. See Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 481 (1974) (explaining that the “general store of knowledge” is so important that the Federal government is willing to pay for it with the grant of a monopoly); \textit{Table 4.7: Judicial Facts and Figures 2010, UNITED STATES COURTS}, http://www.uscourts.gov/Statistics/JudicialFactsAndFigures/JudicialFactsAndFigures
At its most basic level, the quid pro quo of the patent system—monopoly in exchange for disclosure—has broken down.\(^{17}\)

The problem is not that software presents a new or insurmountable challenge to the patent system.\(^{18}\) Rather the patent system itself has failed to properly adjust to the field of software programming in the same way it has adjusted to other new fields of science.\(^{19}\) This Note argues that the patentable subject matter controversy is not itself the problem, but is instead a symptom caused by inadequate enforcement of disclosure requirements.\(^{20}\) Thus, by repairing what is broken about the rules governing disclosure, the USPTO will only grant software patents that adequately disclose inventions and have narrower claims.\(^{21}\) Those claims will neither restrain innovation nor stifle disclosure.\(^{22}\) Instead, those documents will be meaningful contributions to the public store of knowledge, providing invaluable technical information to future innovators.\(^{23}\) Best of all, the patentable subject matter controversy can be put to rest at last.\(^{24}\)

This Note examines the current controversy over patentable subject matter and charts a course back to reasonable grants of patent protection for software. Part I explores the unique challenges that software poses to the patent system. Part II analyzes the current state of the controversy in the Federal Circuit as seen in the recent case, *CLS Bank International v. Alice Corp.*\(^{25}\) Part III recommends a solution that should place software patentability on a surer footing by repairing the problems with enablement and claim construction that have contributed to the current confusion, and liberalizing the patentable subject matter test, so it does not serve as a trap for the unwary.

---

\(^{17}\) See Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 150–51 (1989) (“In consideration of its disclosure and the consequent benefit to the community, the patent is granted.”).

\(^{18}\) See, e.g., Diamond v. Chakrabarty, 447 U.S. 303, 309–10 (1980) (holding that a man-made organism is patentable, thus extending patent protection to the then nascent field of biotechnology).

\(^{19}\) See id.

\(^{20}\) See infra Part I.B.

\(^{21}\) See infra Part I.B.3.

\(^{22}\) See infra Part I.B.2.

\(^{23}\) See infra Part I.B.2.

\(^{24}\) See infra Part III.A.

I. PATENT DOCTRINES AND SOFTWARE CHALLENGES

Controversy has surrounded software’s patentability since the beginning of the digital era.\(^{26}\) In *Gottschalk v. Benson*, the first case to address software patentability, the Supreme Court excluded software completely from the patent system.\(^{27}\) Years later, the Court allowed software patents in *Diamond v. Diehr*, but only for software combined with a machine that produced a physical result.\(^{28}\) The Federal Circuit finally broadly permitted software patents in the *State Street Bank* decision, which allowed patents on machines and processes that produced a numerical, rather than physical, result.\(^{29}\) A few years later, the Supreme Court reined in that doctrine in *Bilski v. Kappos*.\(^{30}\)

This section describes the larger framework of the controversy by (1) discussing the foundational ideas and goals of the patent system,\(^{31}\) (2) outlining the particular challenges that software presents to the patent system,\(^{32}\) and (3) discussing the genesis of the patentable subject matter controversy in the courts through the historic telegraph case, *O’Reilly v. Morse*.\(^{33}\)

### A. First Principles of Patent Law

Fundamentally, the US patent system is an outgrowth of the Intellectual Property (IP) Clause of the US Constitution, which empowers Congress “[t]o promote the [p]rogress . . . [o]f useful [a]rts, by securing for limited [t]imes to . . . [i]nventors the exclusive [r]ight to their . . . [d]iscoveries.”\(^{34}\) Scholars generally recognize that the patent system must provide two incentives: (1) an incentive to invent and (2) an incentive to disclose.\(^{35}\) The incentive-to-invent rationale recognizes that inventors will have to expend resources to invent new

---

27. See id.
31. See infra Part I.A.
32. See infra Part I.B.
33. See infra Part I.C.
34. U.S. CONST. art. I, § 8, cl. 8.
Once invented, competitors may free ride and produce the invented product at a lower cost without investing in expensive research. If patent holders are unable to artificially inflate prices, prospective inventors may be unwilling to fund research in the pursuit of new inventions.

The incentive-to-disclose rationale addresses the problem that trade secret protection poses to the progress of science. When inventors create new devices, they may not reveal how the invention works, and instead conceal its inner workings as a trade secret. For the inventor, this gives a de-facto monopoly that lasts until other innovators are able to reverse engineer or independently develop the same technology. If these efforts are successful, the inventor's monopoly ends. If these efforts fail, inventors can keep their inventions secret and potentially exploit them indefinitely. In either case, knowledge of how the invention works will not enter the public domain; instead, only the company willing to pay to reverse engineer will obtain that knowledge. The incentive to disclose is intended to prevent costly duplicative research. After receiving a patent, the patentee knows that his exclusive right to an invention is secure for a term of years. In exchange, the inventor grants to the public a disclosure of how the invention works, thus eliminating the need for duplicative research.

To ensure the public actually receives sufficient information to prevent that duplicative research, US law requires inventors to

37. See id. at 1024–25.
38. See id. at 1025–26.
39. See Bonito Boats, 489 U.S. at 151 (“[T]he ultimate goal of the patent system is to bring new designs and technologies into the public domain through disclosure.”); Cont'l Paper Bag Co. v. E. Paper Bag Co., 210 U.S. 405, 424 (1908) (“and it was further said in [Am. Bell Tel. Co.] that the inventor could have kept his discovery to himself; but, to induce a disclosure of it, Congress has . . . guaranteed to him an exclusive right to it for a limited time” (citing United States v. Am. Bell Tel. Co., 167 U.S. 249 (1897))); Eisenberg, supra note 35, at 1028–30.
41. See id. at 476 (discussing the availability of reverse engineering to uncover trade secret information).
42. See id.
43. See id.
44. See id.
45. See Eisenberg, supra note 35, at 1028–30.
47. See Eisenberg, supra note 35 at 1028–30.
describe their inventions with sufficient detail to enable a person having ordinary skill in the art (PHOSITA) to make or use the patented invention. This requirement is known as “enablement” and is codified at 35 U.S.C. § 112(a). A “commensurability” requirement compels inventors to match the scope of the disclosure with the scope of the claims. That is, an enabling patent specification must disclose enough information for the PHOSITA to practice the entire scope of the invention as described in the claims.

Disclosure is only one of many requirements necessary to obtain a patent. First, the invention must fall into one of the categories of patentable subject matter and be useful. Second, the invention must be novel. Third, the claimed invention must not be obvious to a PHOSITA in light of technology already known (i.e., prior art). Finally, the applicant must meet several specific disclosure requirements: the application must comply with the written description requirement, be enabled (as discussed above), and disclose the best mode of practicing the invention.

One of the threads that weave these requirements together is the concept of the PHOSITA. The PHOSITA appears in many contexts, but three are particularly important as applied to software: claim construction, enablement, and non-obviousness. Though

---

50. Id.
51. See In re Wright, 999 F.2d 1557, 1561 (Fed. Cir. 1993); MPEP § 2164.08 (8th ed. Rev. 9, Oct. 2012).
53. See id. § 101.
54. See id. §§ 101–102.
55. See id. § 103.
56. See id. § 112.
57. See id. § 103 (codifying the PHOSITA concept); Hotchkiss v. Greenwood, 52 U.S. 248, 267 (1851) (using the phrase “ordinary mechanic acquainted with the business” as the standard for non-obviousness).
58. See Phillips v. AWH Corp., 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc) (explaining that the meaning of claim terms “is the meaning that the term would have to a person of ordinary skill in the art at the time of the invention” (citing Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc., 381 F.3d 1111, 1116 (Fed. Cir. 2004); Home Diagnostics, Inc. v. LifeScan, Inc., 381 F.3d 1352, 1358 (Fed. Cir. 2004); Ferguson Beauregard/Logic Controls v. Mega Sys., LLC, 350 F.3d 1327, 1338 (Fed. Cir. 2003); PC Connector Solutions LLC v. SmartDisk Corp., 406 F.3d 1359, 1363 (Fed. Cir. 2005); Schering Corp. v. Amgen Inc., 222 F.3d 1347, 1353 (Fed. Cir. 2000))).
59. See Auto Techs. Int’l, Inc. v. BMW of North America, Inc., 501 F.3d 1274, 1282 (Fed. Cir. 2007) (“[T]he ‘enablement requirement is satisfied when one skilled in the art, after reading the specification, could practice the claimed invention without undue experimentation.” (quoting AK Steel v. Sollac, 344 F.3d 1234, 1244 (Fed. Cir. 2003))).
the level of skill of a PHOSITA is the same, there are slight variations in how creative the PHOSITA may be, as well as how much of the prior art a PHOSITA knows. The Federal Circuit has outlined a broad framework for determining the relevant level of skill for the PHOSITA, but there is no unified standard dictating precisely when in the course of litigation a court should determine that level of skill. The PHOSITA also serves another key role in the US patent system by adapting each of the various requirements to different technology fields. It is for this reason that the PHOSITA is a key concept for those attempting to understand the many problems unique to software patents.

**B. The Problem of Software**

Software presents several challenges to many of the fundamental presuppositions of patent law. First, it belongs to the class of inventions that are not particularly susceptible to reverse engineering, thus allowing for strong trade secret protection that weakens the incentives to seek patent protection. Second, even issued software patents do not effectively disclose how the claimed software actually functions. In a string of cases, the Federal Circuit has defined the software PHOSITA to be someone of extraordinary skill, thus permitting applicants to obtain a patent in exchange for minimal disclosure. Third, as a result of these two problems, the

---

60. See KSR Int'l Co. v. Teleflex Inc., 550 U.S. 398, 418 (2007) (“Often, it will be necessary for a court to look to[, among other things,] the background knowledge possessed by a person having ordinary skill in the art . . . in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.”).


62. See, e.g., Daiichi Sankyo Co. v. Apotex, Inc., 501 F.3d 1254, 1256 (Fed. Cir. 2007) (Currently, the standard for determining the relevant level of skill of a PHOSITA is a six-factor test that includes: “(1) the educational level of the inventor; (2) type of problems encountered in the art; (3) prior art solutions to those problems; (4) rapidity with which innovations are made; (5) sophistication of the technology; and (6) the educational level of active workers in the field.” (quoting Envtl. Designs, Ltd. v. Union Oil Co., 713 F.2d 693, 696 (Fed. Cir. 1983))).

63. See Burk & Lemley, *supra* note 61, at 1185–89.

64. See infra Part I.B.

65. See infra Parts I.B.1–3.


68. See Northern Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 942 (Fed. Cir. 1990) (quoting *In re Sherwood*, 613 F.2d 809, 817 n.6 (C.C.P.A. 1980)) (saying that the creation of software is a “mere clerical” task for a skilled programmer); Burk & Lemley, *supra* note 61, at 1162–63.
USPTO is unable to effectively examine future software-patent applications. Effective examination of patents requires that the examiners know the general state of the art so they can determine whether patents are new and non-obvious. Most of this knowledge comes from prior patents and published patent applications. Thus, when the USPTO issues poorly enabled patents, it creates a feedback loop, preventing the office from effectively examining future applications.

1. Reverse Engineering

The USPTO is hardly alone in struggling to understand the landscape of modern software development. For this reason, scholars have spilled considerable ink defending the right of non-patentees to reverse engineer software to uncover its functionality. Reverse engineering is the least costly way for competitors to unlock the secrets of patented software and understand the functionality that a patent specification may only suggest.

To begin, some background on the technical details of software may be helpful. Software exists in two basic forms: source code and object code. The source code consists of human-readable text that
explains the functionality of the software. To run the code, a compiler must convert the source code to create computer-readable object code. That process strips away commentary and other distinguishing features of the source code that are critical for human programmers and irrelevant to the computers that run the object code. Usually programmers distribute only the object code while preserving the source code as a trade secret. This renders the original source code unavailable in practice, leaving competitors with only the object code to analyze.

Object code can be decompiled back into a form of source code; however, that decompiled code lacks the developers’ commentary and does not reflect the structure of the original source code. For example, compilation usually cannot even reveal which high-level programming language (like C++, Objective-C, or C#) the original programmer used when authoring the source code. Thus, while the decompiled result may be useful, it is probably nothing like the original source code. Even so, any uncovering of functionality still requires that the prospective reverse engineer have access to the original object code in some way.

The recent trend towards “cloud-based” computing may put a large percentage of code completely out of reach. Cloud computing is a computing model that allows “ubiquitous, convenient, on-demand network access” to services, platforms, and network infrastructures. By definition, this requires multiple computers working across...
networks.\footnote{See Marc Aaron Melzer, Copyright Enforcement in the Cloud, 21 FORDHAM INTL. PROP. MEDIA & ENT. L.J. 403, 404 (2011).} Examples of these sorts of cloud-computing applications include Google Search, Facebook, and web-based email applications.\footnote{Id. at 405.} A prospective reverse engineer without access to every computer involved in the application will likely not be able to adequately observe the system’s functionality, thus making it nearly impossible to reverse engineer the software.\footnote{Id.; see Gutierrez, supra note 87, at 603.} This effectively eliminates reverse engineering as a feasible method of uncovering software functionality from a wide- and fast-growing group of software technologies.\footnote{See Christopher Soghoian, Caught In The Cloud: Privacy, Encryption, And Government Back Doors In The Web 2.0 Era, 8 J. TELECOMM. & HIGH TECH. L. 359 (2010); CSA & ISACA, CLOUD COMPUTING MARKET MATURITY STUDY RESULTS 1, available at https://downloads.cloudsecurityalliance.org/initiatives/collaborate/isaca/2012-Cloud-Computing-Market-Maturity-Study-Results.pdf (last visited Oct. 18, 2013) (“Cloud computing . . . is at the point of advancing from infancy to growth and is reaching a level of maturity at which enterprises can benefit greatly by adopting cloud infrastructure, platform or software service offerings.”).}

Reverse engineering software also faces numerous legal hurdles. Copyright law, patent law, state trade secret law, and contract law all erect barriers to reverse engineering.\footnote{See Cohen, supra note 66, at 1095.} For example, software developers often copyright both the object code and the source code.\footnote{See Litman, supra note 77, at 200.} Earlier types of copyrighted works (for example, books and visual art) are not generally targets of reverse engineering, so there was originally some concern that reverse engineering, which often involves making intermediate copies, could constitute copyright infringement.\footnote{See, e.g., Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527 (9th Cir. 1992).} The Ninth Circuit put those fears to rest in Sega Enterprises Ltd. v. Accolade, Inc., holding that the intermediate copying was a form of fair use.\footnote{See id. This holding has been consistently followed in similar cases. See, e.g., Sony Computer Entm’t, Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000).} This victory, however, may be short-lived in light of the Digital Millennium Copyright Act’s (DMCA) antircumvention provisions.\footnote{See Digital Millennium Copyright Act, 17 U.S.C. § 1201 (2012).} The DMCA codifies the holding in Sega, but it only permits software to be reverse engineered “for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program.”\footnote{Id. § 1201(f)(1).} Because “interoperability” is the only permitted purpose of reverse engineering, this implies that the DMCA
excludes reverse engineering in order to understand how a program works or creates a competing program.99

A separate provision of the DMCA specifically outlaws devices that are “primarily designed or produced for the purpose of circumventing protection afforded by a technological measure that effectively protects a right of a copyright owner.”100 This, read literally, would almost certainly prohibit any form of software intended to aid in the decompilation process or otherwise assist in reverse engineering software.101 For these reasons, the § 112 disclosure requirements of patent law, along with the incentive to disclose, may be the only means by which to determine with any clarity how software operates.102 Patent law therefore presents a unique opportunity to mandate disclosing software’s inner workings.103

2. The Software PHOSITA

The definition of a software PHOSITA as a person of extraordinary skill unfortunately leads to minimal disclosure of how software inventions work, upsetting the standard balance between protection and disclosure.104 For example, in Northern Telecom, Inc. v. Datapoint Corp., the Federal Circuit described a programming language as “simply a highly structured language” and that the “conversion of a complete thought . . . into a language a machine understands is necessarily a mere clerical function to a skilled programmer.”105 As mentioned before, the PHOSITA plays a critical role in tailoring the patent statutes to specific industries.106 In some industries, once the Federal Circuit determines the level of skill of the PHOSITA in a particular industry, that level of skill tends to stick.107

Part of this is a result of the Markman-Cybor claim-construction regime that allows the Federal Circuit to review

99. See id.
100. Id. § 1201(b)(1)(A).
102. See supra notes 48–51 and accompanying text.
103. See id.
104. See Northern Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 942 (Fed. Cir. 1990) (quoting In re Sherwood, 613 F.2d 809, 817 n.6 (C.C.P.A. 1980)).
105. Id. (emphasis added) (quoting In re Sherwood, 613 F.2d 809, 817 n.6 (C.C.P.A. 1980)).
107. See Burk & Lemley, supra note 61, at 1196.
claim constructions de novo.\textsuperscript{108} In \textit{Markman v. Westview Instruments}, the Supreme Court held that claim construction was a question of law for a judge.\textsuperscript{109} In \textit{Cybor v. FAS Technologies}, the Federal Circuit held that trial court claim constructions were entitled to no deference and reviewable de novo.\textsuperscript{110} These precedents rely on the idea that construing the words of a legal document is a task that the court is uniquely suited to, such as in contract law.\textsuperscript{111} A side effect of granting de novo review is that PHOSITA skill levels in prior cases tend to influence later cases, even though the PHOSITA’s level of skill is an underlying question of fact.\textsuperscript{112}

The Markman-Cybor regime is a controversial one that the Federal Circuit has revisited on two occasions.\textsuperscript{113} First, in its landmark \textit{Phillips v. AWH} decision, the Federal Circuit included the issue in its order granting rehearing en banc, but refused to address it.\textsuperscript{114} The Federal Circuit returned to the issue in \textit{Lighting Ballast Control LLC v. Phillips Electric}, ultimately reaffirming the Markman-Cybor regime.\textsuperscript{115} In \textit{Lighting Ballast}, the district court issued a Markman order construing the term “voltage source means.”\textsuperscript{116} The patentee argued that “voltage source means” would be understood by a PHOSITA to mean a rectifier, which is the only type of device that could fulfill the other requirements of the claim.\textsuperscript{117} In the trial court, the patentee presented expert testimony and other extrinsic evidence to show that this was the correct construction.\textsuperscript{118}
The original panel reversed without the benefit of hearing that testimony, however, and held the claim invalid as indefinite.\(^\text{119}\) When the Federal Circuit took the case en banc, it reaffirmed the Markman-Cybor framework, meaning that the software PHOSITA will likely continue to have extraordinary skill.\(^\text{120}\)

This attitude likely explains the current problems with software patents.\(^\text{121}\) Defining a PHOSITA as one with an extremely high level of skill effectively eviscerates the enablement requirements under § 112.\(^\text{122}\) In one of the most extreme examples, \textit{In re Dossel}, the Federal Circuit found that a software patent that neither used the word “computer” nor contained any computer code at all satisfied the enablement requirement of § 112.\(^\text{123}\) As a result, software patents that do issue tend to have extraordinarily broad claims and little meaningful disclosure, if any.\(^\text{124}\)

The enablement requirement differs for varying technological fields, based on the predictability of the art.\(^\text{125}\) In some fields, lower standards of enablement make sense.\(^\text{126}\) Among “predictable” fields of invention, like mechanical devices, the specification only needs to disclose a few possible ways of practicing the invention, whereas in “unpredictable” fields, like biotech and pharmaceuticals, the specification must include in-depth disclosure of every possible means of practicing the invention.\(^\text{127}\) Software, however, appears to be the only field that exists outside this spectrum; a patent can meet the enablement requirement without disclosing even a single embodiment.\(^\text{128}\) \textit{In re Dossel} demonstrates that the patent system’s basic quid pro quo—the grant of a monopoly to the inventor in exchange for disclosure to the public—has completely broken down.\(^\text{129}\)

\(^{119}\) \textit{Id.} at 992.

\(^{120}\) \textit{See id.} at 1168.

\(^{121}\) \textit{See id.}

\(^{122}\) \textit{See id.} at 1196.

\(^{123}\) \textit{See In re Dossel}, 115 F.3d 942, 946 (Fed. Cir. 1997).

\(^{124}\) \textit{See, e.g.}, Bilski v. Kappos, 130 S. Ct. 3218, 3223–24, 3231 (2010) (explaining that the claims at issue were so broad that they did not require a computer, and constitute a patent on the concept of hedging).


\(^{126}\) \textit{See id.}

\(^{127}\) \textit{See Burk & Lemley, supra note 61, at 1196; Seymore, supra note 125, at 136–39.}

\(^{128}\) \textit{In re Dossel}, 115 F.3d at 946–47.

\(^{129}\) \textit{See id.}
3. Examination Issues

In 1965, a presidential commission on the patent system concluded that the USPTO was unable to examine applications for software patents because it had no classification system or sufficient prior-art files with which to properly examine the applications.\textsuperscript{130} Since then, little has changed.\textsuperscript{131} The lack of sufficient prior-art databases for the purposes of examining software patents is a major driver of poor patent quality.\textsuperscript{132} This problem is not entirely the USPTO's fault.\textsuperscript{133} Unlike advances in fields such as biotechnology, software improvements rarely appear in peer-reviewed journals, conference presentations, or other media that traditionally establishes prior art in other fields.\textsuperscript{134} There are several professional associations that publish scholarly materials, primarily the Association for Computing Machinery (ACM)\textsuperscript{135} and the Institute of Electrical and Electronics Engineers (IEEE).\textsuperscript{136} But the nature of the software industry dictates that business, and not academia, drives innovation.\textsuperscript{137} Generally, businesses simply incorporate their innovations into products without documentation or only describe them in user manuals that are frequently unavailable to examiners.\textsuperscript{138} While these materials would certainly be useful, the most effective form of prior art for software patents is probably original source code with the original developer comments.\textsuperscript{139}

That source code, however, is almost never available for two reasons.\textsuperscript{140} First, the owners of the source code, containing the

\textsuperscript{131}.  See Cohen & Lemley, supra note 64, at 12–13.
\textsuperscript{132}.  See Cohen, supra note 74, at 1177–80; Cohen & Lemley, supra note 66, at 12–13.
\textsuperscript{133}.  See Cohen, supra note 74, at 1177–80.; Cohen & Lemley, supra note 66, at 42.
\textsuperscript{134}.  See Cohen, supra note 74, at 1177–78; Cohen & Lemley, supra note 66, at 42.
\textsuperscript{137}.  See Cohen & Lemley, supra note 66, at 42.
\textsuperscript{138}.  Cohen, supra note 74, at 1177–78.
\textsuperscript{139}.  See Burk & Lemley, supra note 61, at 1166 n.46 ("Professor Randall Davis of MIT summed it up at the National Research Counsel in 1990: "There is almost no way to visualize software. Sure, we have flow charts, we have data-flow diagrams, we have control flow diagrams, and everybody knows how basically useless those are." (quoting Thomas P. Burke, Note, Software Patent Protection: Debugging the Current System, 69 Notre Dame L. Rev. 1115, 1158–60 (1994))).
\textsuperscript{140}.  See infra notes 141–150 and accompanying text.
developer's commentary on the software's functionality, protect the code as a trade secret.\textsuperscript{141} In some cases, independent open-source developers have disseminated source code more widely in the interest of making code more accessible to third parties.\textsuperscript{142} While open-source code may be available online, and thus to patent examiners, the innovative content of open-source code often lags far behind its commercial competitors.\textsuperscript{143}

Second, even though copyright registration could potentially offer an avenue for disclosure, the Copyright Office currently does not require developers to provide a complete copy of their code.\textsuperscript{144} Most software developers do file copyright registrations, which ordinarily require that the registrant deposit copies of the material with the Copyright Office.\textsuperscript{145} Unfortunately, the Copyright Office does not require the deposit of a complete copy, but only requires an excerpt of the source code.\textsuperscript{146} While the Copyright Office could require the deposit of a complete copy, the size of many software programs would make storage cost prohibitive, as many commercial software packages contain millions of lines of code.\textsuperscript{147} Further, to zealously guard their code, companies could forgo copyright registration and only register before they enforce their copyrights against possible infringers.\textsuperscript{148} Preventing such a workaround would be no small task, as the Berne Convention, to which the United States is a signatory, prohibits the

\begin{itemize}
\item 143. For example, the Secure Sockets Layer (SSL) encryption technology, which enables servers and web browsers to communicate securely, was developed by Netscape in 1996. See A. Freier et al., \textit{The Secure Sockets Layer (SSL) Protocol Version 3.0, Internet Engineering Task Force} (August 2011), http://www.rfc-base.org/txt/rfc-6101.txt. However, the first open-source alternative, OpenSSL, was not available until the end of 1998. Tarballs, OPENSSL (Oct. 3, 2013, 8:57 PM), http://www.openssl.org/source/ (showing a timestamp of the first version in December of 1998).
\item 144. See 37 C.F.R. § 202.20(c)(2)(vii) (2012).
\item 145. See 17 U.S.C. § 407 (2012) (“[T]he owner of copyright . . . shall deposit, within three months after the date of such publication . . . two complete copies of the best edition . . .”).
\item 146. See 37 C.F.R. § 202.20(c)(2)(vii) (2012) (requiring only “identifying” portions of the program to be deposited, including either the first and last twenty-five pages of code, or, if the source code is held as a trade secret, only the “page or equivalent unit containing the copyright notice” plus several pages with trade secret materials blocked out).
\item 148. See 17 U.S.C. § 102(a) (2012) (stating that copyright exists once a work of authorship is “fixed in any tangible medium of expression” and thus does not require registration).
\end{itemize}
United States from conditioning protection on registration. Therefore, patent law remains in the best position to provide for public disclosure of software inventions.

C. The Genesis of Patentable Subject Matter Doctrine

For patent law to serve that purpose, software must first be patentable. Whether software is patentable is at least nominally an issue of construing 35 U.S.C. § 101, which states: “Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter . . . may obtain a patent therefor, subject to the conditions and requirements of this title.”

Notwithstanding one minor alteration, the language of this statute has remained fundamentally unchanged since the Patent Act of 1793. Congress has suggested that the four statutory categories—processes, machines, manufactures, and compositions of matter—encompass “anything under the sun that is made by man.” Consistent with this limitation, courts have read into the statute three major exceptions: “laws of nature, physical phenomena, and abstract ideas.” The Supreme Court has typically explained that these exclusions are intended to prevent patents that would “preempt” too much follow-on innovation. Simply put, patents for inventions in these excluded categories would not serve the constitutional aim of encouraging innovation but would instead prevent the use of discoveries that are the “basic tools of scientific and technological work.”

Of the three exclusions from patentable subject matter, patents on software often conflict with the abstract-idea exception. Method

150. See supra Part I.B.
156. See id.
157. See generally Peter S. Menell, Forty Years of Wondering in the Wilderness and No Closer to the Promised Land: Bilski’s Superficial Textualism and the Missed Opportunity to
claims that only specify an algorithm itself fall plainly within the abstract-idea exception, particularly within the well-defined category of mathematical algorithms. Claims that attempt to capture software algorithms as tied to a “general-purpose computer” or as a “software product” have also run into difficulty because courts do not view those limitations as meaningful. In fact, courts often focus their § 101 analysis on the method claim, and, after finding it directed towards non-statutory subject matter, invalidate the remaining media and system claims as a matter of course. Courts have wrestled with the question of software for nearly forty years, yet there is still no clear guidance to patent practitioners as to whether or when software is patent eligible.

The first case to squarely address software was *Gottschalk v. Benson*, a case in which the Supreme Court held that a software method for converting binary numbers was not patent eligible. Since the decision to exclude software from patent protection in *Benson*, the Supreme Court and Federal Circuit have alternated between two general approaches to the patent eligibility of software. The first approach is exemplified in the classic *Benson-Flook-Diehr* trilogy that explored whether software had to be tied to a machine or confined in application to be potentially patentable. This approach was based on the fear that software patents would preempt too much future research because they were abstract ideas. The Supreme Court only allowed software patents in *Diehr*, and only then when

---

*Return Patent Law to Its Technology Mooring, 63 STAN. L. REV. 1289 (2011) (describing the history of software’s questionable patentability up to *Bilski*).*

158. See *Benson*, 409 U.S. at 71–72.

159. SiRF Tech., Inc. v. Int’l Trade Comm’n, 601 F.3d 1319, 1332–33 (Fed. Cir. 2010) (“[F]or the addition of a machine to impose a meaningful limit on the scope of the claim, it must play a significant part in permitting the claimed method to be performed, rather than function solely as an obvious mechanism for permitting a solution to be achieved more quickly . . . .”); see *Diamond v. Diehr*, 450 U.S. 175, 187 (1981) (finding a computer system in conjunction with a rubber-curing press patent eligible).


163. See supra notes 151–56 and accompanying text.


165. See *Benson*, 409 U.S. at 71–72.
computers were incorporated into other machines. The second approach originates in the *State Street Bank* decision. In that case, the Federal Circuit concluded that inventions were patentable if they resulted in a “useful, concrete, and tangible result.” *State Street Bank* allowed for much broader software inventions and did not require such a strong link between a machine and its software.

Out of a concern that the *State Street Bank* regime permitted inventors to obtain too many patents on software and business-method patents, the Federal Circuit returned to the *Benson-Flook-Diehr* preemption approach in *In re Bilski* by formulating the machine-or-transformation test and rejecting the “useful, concrete, and tangible result” test. This line reflects the current state of software-patent jurisprudence. Even though the Supreme Court rejected the machine-or-transformation test as the sole test for patentable subject matter, the Court endorsed the Federal Circuit’s preemption approach, discussed below.

As this discussion illustrates, the foundations of software-patent eligibility rest on shifting ground. *Benson* and its progeny draw heavily from *O'Reilly v. Morse*, an 1861 case involving Samuel Morse, the inventor of the telegraph. *Benson, Flook, Diehr, Bilski, and Mayo* all cite *Morse* as the genesis of the preemption view of patentable subject matter. In that case, Samuel Morse attempted to obtain a patent on “the use of the motive power of the electric or galvanic current . . . however developed for marking or printing intelligible characters, signs, or letters, at any distances, being a new...

---

166. See Diehr, 450 U.S. at 192–93

167. State St. Bank & Trust Co. v. Signature Fin. Grp., 149 F.3d 1368, 1373 (Fed. Cir. 1998) (citing In re Alappat, 33 F.3d 1526, 1544 (Fed. Cir. 1994)).

168. *State Street Bank*, 149 F.3d at 1373.

169. See id.


171. See Mayo Collaborative Servs. v. Prometheus Labs., Inc., 132 S. Ct. 1289, 1298 (2012) (expressing concern that the claims would preempt a natural law, and thus are unpatentable).


174. See O’Reilly v. Morse, 56 U.S. (15 How.) 62, 112 (1853); infra notes 176–81 and accompanying text.

application of that power . . . ."176 In rejecting the claim, the court reasoned:

If this claim can be maintained, it matters not by what process or machinery the result is accomplished. For aught that we now know some future inventor, in the onward march of science, may discover a mode of writing or printing at a distance by means of the electric or galvanic current, without using any part of the process or combination set forth in the plaintiff’s specification.177

In essence, the Court reasoned that if Morse obtains a patent on the challenged claim, it would “preempt” all other forms of electronic communication.178 The Court, however, rejected the claim because it was not enabled, and not because it was patent ineligible.179 In theory, if Morse’s disclosure would have enabled a PHOSITA to implement every possible means of electronically “marking or printing” messages at a distance, the patent would have enabled the claim, and the court should have upheld the patent.180 In summary, disclosing one method of electronic communications does not enable a patent that can preempt the underlying idea of electronic communication.181

Seeing this connection, some have argued that patent eligibility should be of no concern to the courts at all, but that the other substantive requirements are adequate to disqualify overly preemptive patents.182 While such an extreme position is certainly too heavy-handed, the link drawn between enablement and patentable subject matter is a crucial one.183 Patentable subject matter serves a valuable role as a last firewall defending the patent system from overly broad, innovation-hindering patent claims when enablement and the other requirements of patentability have failed to do so.184 In fact, the Federal Circuit’s citations to Morse support this understanding: the court cites the same passage from Morse to

176. Morse, 56 U.S. (15 How.) at 112.
177. Id. at 113.
178. See id.
179. See id. (“In fine he claims an exclusive right to use a manner and process which he has not described and indeed had not invented, and therefore could not describe when he obtained his patent.”).
180. See id. at 112.
181. See id. at 113.
182. See generally Michael Risch, Everything is Patentable, 75 TENN. L. REV. 591 (2008) (arguing that courts should allow patents on anything that meets the other conditions of patentability under §§ 102, 103, and 112).
183. See supra notes 151–66 and accompanying text.
184. See generally Cohen & Lemley, supra note 66 (discussing how patentable subject matter constrains claim scope).
support both the preemption foundation of patent eligibility and the concept of commensurability.185

II. CLS BANK V. ALICE CORP.

The controversy over patentable subject matter continues, however, as demonstrated in CLS Bank v. Alice Corp.186 CLS Bank addressed a validity challenge to a patent on a computerized stock-trading platform for use by mutual funds.187 The plaintiff was an Australian company that held four US patents on an automated method for settling large trades in international-currency markets.188 The defendant was an international currency-trading platform operating in foreign-exchange markets to mitigate settlement risks between its customers and members.189 Those members included the central banks of seventeen nations and seventy-two worldwide banking institutions.190

The first opinion in the case was a split decision.191 The CLS Bank I majority attempted to minimize the importance of the § 101 analysis by permitting the invalidation of patents if they covered clearly unpatentable subject matter, giving trial courts discretion to invalidate patents on other grounds, rather than solely treating § 101 as a threshold.192 The CLS Bank I dissent applied § 101 by considering natural phenomena to be within the prior art, and then performing a brief non-obviousness analysis in light of the phenomena.193

185. E.g., Ariad Pharm., Inc. v. Eli Lilly & Co., 598 F.3d 1336, 1359 (2010) (citing to Morse as support for the enablement and written description requirements of § 112); Carnegie Mellon Univ. v. Hoffmann-La Roche Inc., 541 F.3d 1115, 1121–22 (2008) (“It has long been the case that a patentee ‘can lawfully claim only what he has invented and described, and if he claims more his patent is void.’”) (quoting O’Reilly v. Morse, 56 U.S. (15 How.) 62, 121 (1853)).
186. CLS Bank III, 717 F.3d 1269, 1274 (Fed. Cir. 2013) (en banc).
187. See id.
189. See CLS Bank I, 768 F. Supp. 2d at 223.
192. See id. at 1347–52.
193. See id. at 1356–59.
In light of the need for further guidance, and in the hope of arriving at a unified standard in this area, the Federal Circuit agreed to rehear the CLS Bank I case en banc. Measured against those goals, the CLS Bank II opinions are unfortunately a complete failure. An equally divided Federal Circuit upheld the trial court’s decision in a mere fifty-eight-word per curiam decision. What followed in over one hundred pages were five separate opinions plus an “Additional Reflection” filed by Chief Judge Rader. Those pages present multiple approaches to determining if an invention is patent eligible under § 101. Among the opinions, two general approaches to the patentable subject matter inquiry emerged: one from the primary concurrence, written by Judge Lourie, and one from the primary dissent, authored by Chief Judge Rader. The main distinction on which the varying opinions focused was whether system, method, and “computer implemented” claims all rise and fall together. The primary concurrence reasoned that they did, and thus were all invalid. The primary dissent argued that computer-implemented claims contained sufficient additional limitations to be patent eligible, specifically because those limitations tied the claims to a machine, satisfying the machine-or-transformation test.

A. Lourie’s Concurrence—Coextensivity

Judge Lourie’s concurrence was the primary opinion supporting the trial court’s decision that the claims in CLS Bank were not patent eligible. The opinion returns the § 101 analysis to its preemption roots, essentially asking if there is a “practical likelihood
of a claim preempting a fundamental concept.”

To apply this concept, the court begins by asking whether the claim would possibly preempt some abstract idea. If the reviewing court uncovers such an abstract idea, it analyzes the claim to determine whether, once the abstract idea is removed from the claim, it contains any other limitations that would keep the claim from “cover[ing] the full abstract idea itself.”

In essence, the court determines the scope of the abstract idea and then compares the claim to that scope to determine if they are the same. If both the idea and the claim are coextensive, the claim is invalid. Assuming meaningful limitations do exist, those limitations constitute the “inventive concept” that renders the claim patent eligible.

A limitation only distinguishes a patent-eligible claim from an abstract idea if the limitation is meaningful, which the opinion defines as something that is not “merely tangential, routine, well-understood, or conventional, or in practice fails to narrow the claim relative to the fundamental principle therein.”

The most obvious examples of meaningless limitations are the claims rejected by the Supreme Court in *Mayo Collaborative Services v. Prometheus Laboratories*. In that case, the claims recited steps of “administering” a drug, and “determining” the level of a particular metabolite. What rendered these limitations meaningless was that they were “necessary to every practical use” of the natural law underlying the claim. In other words, the scope of the claim and the scope of the underlying natural law were effectively the same.

Applying these rules, the primary concurrence would hold the claims to be patent ineligible. As to the method claims, which recite no machine or system, the court reasoned that the mere concept of the trading platform could not be patented because it is “untethered from
any real-world application.” As to the claims that appear to recite computers, or require computer implementation, the court ruled that the claims were patent ineligible because there was “no specific or limiting recitation of essential, . . . or improved computer technology” and that the limitations were indistinguishable from the limitations in Mayo. Interestingly, the mere computer implementation of a method in such a way that the computer lends “speed or efficacy” to the method does not qualify it for patent protection. This creates a conundrum in which some computer-implemented inventions that would otherwise satisfy the machine-or-transformation test may not be patent eligible.

B. Rader’s Dissent

Chief Judge Rader’s dissent did not dispute the patent ineligibility of the system or method claims but fundamentally disagreed with the concurrence regarding whether the computer-implemented claims were patent ineligible. Drawing from In re Alappat, the dissent recognized that “a computer without software collects dust, not data,” and thus the combination of a general-purpose computer with software creates a new, patent-eligible machine. Further, it recognized that Bilski rejected the machine-or-transformation test as too restrictive. Bilski did not imply that inventions that satisfy that test are of questionable patentability, but instead that inventions that fail it may yet be patentable. Thus, it would be odd to invalidate a computer-implemented claim that can only be embodied in a physical, tangible machine. Further, the dissent recognized the driving tension in patent law’s discomfort with software patents—that any method implemented in software could be implemented in clearly patentable hardware.

216. Id. at 1286.
217. Id. (citations omitted).
218. Id. (“Unless the claims require a computer to perform operations that are not merely accelerated calculations, a computer does not itself confer patent eligibility.”).
219. See id.
220. See CLS Bank III, 717 F.3d at 1305 (Rader, C.J., concurring in part and dissenting in part).
221. See id.
222. Id.
223. See id.
224. See id.
225. See id. at 1306; see also In re Alappat, 33 F.3d 1526, 1569–70 (Fed. Cir. 1994) (Newman, J., concurring) (discussing hardware/software equivalence), abrogated by In re Bilski, 545 F.3d 943 (Fed. Cir. 2008); supra notes 178–80 and accompanying text.
The dissent also took issue with the concurrence’s characterization of some limitations as meaningless. It argued that any claim can be simplified down to an abstract idea, and thus such an approach could invalidate scores of patents. Simply put, “a court cannot go hunting for abstractions by ignoring the concrete, palpable, tangible limitations of the invention the patentee actually claims.” When it comes to computer-implemented inventions, the standard should be far simpler: “[W]here the claim is tied to a computer in such a way that the computer plays a meaningful role in the performance of the invention, it should be patent eligible.

Applying that standard to the patents in suit, the dissent concluded that wherever the claim required a computer, the claim recited patent-eligible subject matter. Further, where flowcharts accompanied the computer implementation described in the specification to the method, as here, labeling the claim an “abstract concept” effectively “wrenches all meaning from those words.” As to the method claims, which did not recite a computer and encompass the method of currency trading no matter the application, the dissent agreed that they were not patent eligible.

III. NEEDED REFORMS AT THE USPTO AND IN THE COURTS

The continued divisions among the judges of the Federal Circuit show that the patent system is in need of reform. The central aim of the judicial exclusions from patentable subject matter is generally to prevent the preclusion of wide areas of technological investigation from future research. Rather than treat the doctrine as an aberrant extension of the enablement doctrine, the better approach is to view current patentable subject matter doctrine as a backstop for catching excessively preclusive claims. As pioneers created entirely new fields of invention, courts usually did not stand in

226. CLS Bank III, 717 F.3d at 1309 (Rader, C.J., concurring in part and dissenting in part).
227. See id.
228. Id. at 1298.
229. Id. at 1302.
230. See id. at 1305 (“If tying a method to a machine can be an important indication of patent eligibility, it would seem that a claim embodying the machine itself, with all its structural and functional limitations, would rarely, if ever, be an abstract idea.” (citing Diamond v. Diehr, 450 U.S. 175, 187 (1981))).
231. Id. at 1309.
232. See id. at 1312–13.
233. Id.
235. See supra notes 178–81.
the way, but instead recognized innovations in those fields as patentable.\textsuperscript{236} In Diamond v. Chakrabarty, for example, the Supreme Court did not hesitate to extend patent protection to an invented, non-naturally-occurring microorganism.\textsuperscript{237} In fact, it has now become standard practice in the courts to follow the recitation of § 101 with the well-known quotation that Congress intended § 101 to cover “anything under the sun made by man.”\textsuperscript{238}

Considering the entire story of software, the Federal Circuit’s about-face in Bilski is likely a reaction to the excesses of the State Street Bank regime.\textsuperscript{239} Flooded with low-quality patents with exceedingly large scope, the courts likely adopted the machine-or-transformation test as a limiting principle to stem the tide.\textsuperscript{240} But rather than continue to use § 101 as a crude tool against software patents, the USPTO and the courts should undertake reforms to better enforce the other substantive requirements of patentability.\textsuperscript{241} In this way, the USPTO and the courts should be able to limit the sort of broad, abstract claims in State Street Bank, Bilski, and CLS Bank, while providing patent practitioners with clearer guidance and the software community with a new wealth of technical documentation.\textsuperscript{242}

\textbf{A. A Return to State Street Bank}

Reform will likely have to begin in the courts. Specifically, the Supreme Court should return the § 101 analysis to the State Street Bank formulation.\textsuperscript{243} A computer-software patent should be patentable if it produces “a useful, concrete, and tangible result,” which includes the transformation of data.\textsuperscript{244} This recognizes that patentable subject matter excludes abstract ideas not because they are

\begin{footnotesize}
\begin{enumerate}
\setcounter{enumi}{236}
\item See id.
\item See id.
\item See CLS Bank III, 717 F.3d 1269, 1321 (Fed. Cir. 2013) (en banc) (Newman, J., concurring in part and dissenting in part) (“Section 101 is not the appropriate vehicle for determining whether a particular technical advance is patentable . . . .”).
\item See infra notes 279–288 and accompanying text.
\item See CLS Bank III, 717 F.3d at 1285; In re Bilski, 545 F.3d 943, 959 (Fed. Cir. 2008), aff’d sub nom. Bilski v. Kappos, 130 S. Ct. 3218 (2009); State Street Bank, 149 F.3d at 1373.
\item CLS Bank III, 717 F.3d at 1302 (Rader, C.J., concurring in part and dissenting in part); In re Bilski, 545 F.3d at 959 (quoting State Street Bank, 33 F.3d at 1373); State Street Bank, 149 F.3d at 1373 (quoting In re Alappat, 33 F.3d 1526, 1544 (Fed. Cir. 1994)).
\end{enumerate}
\end{footnotesize}
mental, mathematical, or algorithmic, but because they are not useful.\textsuperscript{245} In fact, the abstract-idea exception should be more doctrinally linked to the utility requirement under § 101 than the current “preemption” approach.\textsuperscript{246} Utility is a concept rooted in the very foundations of our patent law, specifically in the basic quid pro quo it embodies: in exchange for a monopoly grant to the inventor, the public receives an enabling disclosure of a useful invention.\textsuperscript{247} As the Supreme Court put it:

Unless and until a process is refined and developed to this point—where specific benefit exists in currently available form—there is insufficient justification for permitting an applicant to engross what may prove to be a broad field.\textsuperscript{248}

Instead of engaging in the murky metaphysical question of whether an invention is an abstract idea, courts should engage in a more basic analysis of whether the invention is the kind that should merit “the public embarrassment of a patent.”\textsuperscript{249} The USPTO thus should only grants patent for which there is a “specific and substantial” utility.\textsuperscript{250} In other words, the invention, as claimed, must provide some “immediate benefit to the public.”\textsuperscript{251} This link makes sense because the utility requirement and the prohibition on abstract ideas rest on the same doctrinal foundation.\textsuperscript{252} Despite the benefit a patent may bring to the public by disclosing the invention for further research, granting a patent would block areas of scientific pursuit without compensating the public for that monopoly.\textsuperscript{253} Further, the utility analysis asks the patentability question far more directly: What benefit does the public secure in exchange for granting this patent?\textsuperscript{254}

\begin{itemize}
\item \textsuperscript{245} See State Street Bank, 149 F.3d at 1373 (holding an invention patentable if it produces a “useful, concrete, and tangible result”).
\item \textsuperscript{246} See 35 U.S.C. § 101 (2012).
\item \textsuperscript{247} See Brenner v. Manson, 383 U.S. 519, 534–35 (1966).
\item \textsuperscript{248} Id.
\item \textsuperscript{250} See In re Fisher, 421 F.3d 1365, 1371 (Fed. Cir. 2005); MPEP § 2107 II.B (8th ed. Rev. 9, Oct. 2012).
\item \textsuperscript{251} Fisher, 421 F.3d at 1371 (quoting Nelson v. Bowler, 626 F.2d 853, 856 (C.C.P.A. 1980)).
\item \textsuperscript{252} Compare id. at 1375–76 (explaining that, if a patent were to be granted on an invention that was not useful, the patent may serve to “block off whole areas of scientific development” (quoting Brenner v. Manson, 383 U.S. 519, 535–36 (1966))), with O’Reilly v. Morse, 56 U.S. (15 How.) 62, 113 (1861) (reasoning that if the claim were to be allowed on an abstract idea, it would preclude future inventors from practicing their inventions).
\item \textsuperscript{253} See Fisher, 421 F.3d at 1375–76.
\item \textsuperscript{254} Id. at 1371.
\end{itemize}
B. Reconsidering the Order of Validity Challenges

Additionally, the Federal Circuit should change the procedural requirements for considering patentable subject matter challenges. Because the courts have considered patentable subject matter a threshold issue, lower courts generally address it before construing the claim language. It is curious that in the Bancorp decision the Federal Circuit did not fault the district court for deciding the § 101 question prior to claim construction but nonetheless felt it necessary to construe the claim itself before proceeding with a § 101 analysis. This reflects the dichotomy the Federal Circuit perceives between patent eligibility and invalidity. Problems under §§ 102, 103, and 112 are all “conditions of patentability” that courts must therefore consider after claim construction. Some commentators have suggested that § 101 should not be considered in litigation at all. The Federal Circuit should minimize this confusion by expressly permitting trial courts to consider §§ 101, 102, 103, and 112 issues together. A patent that fails to satisfy any one of those sections is invalid. Courts could have disposed of many of the patentable

255. CLS Bank III, 717 F.3d 1269, 1284 (Fed. Cir. 2013) (en banc) (discussing whether § 101 challenges are a “threshold test”).

256. See, e.g., Ultramercial, LLC v. Hulu, LLC, 657 F.3d 1323, 1325 (Fed. Cir. 2011) (“This court has never set forth a bright line rule requiring district courts to construe claims before determining subject matter eligibility.”), vacated sub nom. WildTangent, Inc. v. Ultramercial, LLC, 132 S. Ct. 2431 (2012) (mem.).

257. See Bancorp Servs., L.L.C. v. Sun Life Assurance Co. of Canada (U.S.), 687 F.3d 1266, 1273–75 (Fed. Cir. 2012) (endorsing the district court’s decision to dispose of the case on § 101 grounds prior to construing the claims, and subsequently construing the claims on appeal).

258. See id.


260. This view is rooted in 35 U.S.C. § 282, which enumerates the defenses against validity as including failure to comply with § 112, or “on any ground specified in part II as a condition for patentability.” 35 U.S.C. § 282 (2012). Curiously, §§ 102 and 103 are entitled “Conditions for Patentability; novelty” and “Conditions for Patentability; non-obvious subjectmatter” respectively. 35 U.S.C. §§ 102–103 (2012). Section 101, however, is missing the “Conditions for Patentability” prefix, leading to speculation that it was never intended to be a defense at all. 35 U.S.C. § 101. In fact, all patentable subject matter cases prior to State Street Bank were in the prosecution context. See, e.g., Gottschalk v. Benson, 409 U.S. 63 (1972). Only State Street Bank and Mayo raised patentable subject matter in the litigation context, and neither have addressed this issue. See Mayo Collaborative Servs. v. Prometheus Labs., 132 S. Ct. 1289 (2012); State St. Bank & Trust Co. v. Signature Fin. Grp., 149 F.3d 1368, 1370 (1998), abrogated by In re Bilski, 545 F.3d 943 (Fed. Cir. 2008).

261. See CLS Bank III, 717 F.3d 1269, 1284 (Fed. Cir. 2013) (en banc) (Lourie, J., concurring) (“In addition, district courts may exercise their discretion to begin elsewhere when they perceive that another section of the Patent Act might provide a clearer and more expeditious path to resolving a dispute.”) (citing MySpace, Inc. v. GraphOn Corp. 672 F.3d 1250, 1258–62 (Fed. Cir. 2012); Dennis Crouch & Robert P. Merges, Operating Efficiently Post-Bilski by Ordering Patent Doctrine Decision-Making, 25 BERKELEY TECH. L.J. 1673 (2010)).

subject matter cases on other grounds, avoiding the confusing and unnecessary § 101 issues. In fact, the CLS Bank decision makes clear that courts may consider other grounds for invalidating patents prior to determining patent-eligibility issues. The Federal Circuit should go further and establish that option as the norm.

C. Deference to Trial Court PHOSITA Determinations

Likewise, the Federal Circuit must expressly delegate responsibility for determining the relevant level of skill of the PHOSITA to the trial court. This will require overturning, to some degree, the current Markman-Cybor regime. Though Markman-Cybor considers claim construction to be like interpreting a contract, unlike contract law, neither what the inventor intended nor its objective textual meaning determines the legal interpretation of a claim. The proper construction of a claim is the meaning it would have to a PHOSITA at the time of the invention. As noted above, the Federal Circuit’s precedents that consider the software PHOSITA to have an extraordinary level of skill are unrealistic and undermine the goals of the patent system. Claim construction may remain an issue of law for the courts to determine, but the relevant level of skill of the PHOSITA should be a question of fact, subject to Federal Circuit review only for an abuse of discretion. Markman may remain undisturbed, but the court should revisit Cybor and overrule it.

The Federal Circuit may, in fact, be moving in this direction. Critically, the outcome of Lighting Ballast Control depends on a PHOSITA’s interpretation of claim language, as determined by facts presented to the trial court. Thus, the Federal Circuit should

264. CLS Bank III, 717 F.3d at 1284.
265. See id. at 1284 (Lourie, J., concurring).
269. See id.
270. See supra Part I.B.2.
272. See id.
273. Id. at 992.
274. See id. at 988.
overrule Cybor, affording the trial court deference specifically with respect to the issue of the understanding and level of skill of a PHOSITA.\textsuperscript{275} The Federal Circuit has already given the district courts substantial guidance on how to determine that skill level.\textsuperscript{276} As in Lighting Ballast Control, many of these questions are intensely factual in nature, placing the trial court in a better position to assess these complex issues.\textsuperscript{277} If nothing else, such an approach allows the PHOSITA concept to be more flexible over time, adapting to the complexity of the software patent at issue.\textsuperscript{278}

\textbf{D. Reforms at the USPTO}

Once the courts settle these issues, the USPTO should implement complementary reforms.\textsuperscript{279} After reconciling the examination rules with the new rules above, the USPTO should have more flexibility to enforce enablement and best mode requirements in prosecution.\textsuperscript{280} As software patents are subject to more stringent disclosure requirements, the technical information contained in them should steadily improve with enhanced oversight.\textsuperscript{281} Since patents and published patent applications are the most common sources of prior art in other contexts, this should produce a positive feedback loop as software patents are examined for novelty and non-obviousness in light of more detailed prior art patents.\textsuperscript{282}

Further, when those patents are better examined, applying the \textit{State Street Bank} rule to recent software cases would likely result in the same outcomes, but on different grounds.\textsuperscript{283} For instance, the patented method in \textit{Bilski} involved patenting the concept of hedging risk as applied to computers.\textsuperscript{284} Rather than rejecting the patent as directed towards an abstract idea, the court just as easily could have

\begin{itemize}
  \item \textsuperscript{275} \textit{Cf.} Lighting Ballast, 2014 WL 667499, at *1 (affirming \textit{Markman-Cybor})
  \item \textsuperscript{276} \textit{See, e.g.}, Daichi Sankyo Co. v. Apotex, Inc., 501 F.3d 1254, 1256 (2007).
  \item \textsuperscript{277} \textit{Cf.} \textit{Fed. R. Civ. P.} 52(a).
  \item \textsuperscript{278} \textit{See supra} notes 57–64 and accompanying text.
  \item \textsuperscript{279} \textit{See supra} notes 69–72 and accompanying text.
  \item \textsuperscript{281} Specifically, the current requirements should include more than just the currently required block diagrams. \textit{See MPEP} § 2164.06(c) (8th ed. Rev. 9, Oct. 2012).
  \item \textsuperscript{282} \textit{See id.} § 904.02 (discussing common search areas for patent examiners); \textit{supra} notes 69–72 and accompanying text.
  \item \textsuperscript{283} \textit{See Risch, supra} note 167, at 609–13.
  \item \textsuperscript{284} \textit{SeeCLS Bank III}, 717 F.3d 1269, 1286 (Fed. Cir. 2013) (en banc).
\end{itemize}
rejected it on grounds that applying the old idea of hedging risk to computers was a perfectly obvious combination, and thus not patentable under § 103.285 The court even noted that the hedging process described in the patent was “a fundamental economic practice long prevalent in our system of commerce and taught in any introductory finance class,” and thus could easily have formed the basis of a § 103 rejection.286 On the other hand, had Edsger Dijsktra submitted a patent application on his now-famous route-finding algorithm, as used to route cars across cities or information around the Internet such a claim would not be barred under a machine-or-transformation test.287 Instead, in light of its tremendous usefulness, novelty, and non-obviousness, a patent would have issued, protecting Dijsktra’s interests and making truly useful information about the algorithm made available to the public.288

IV. CONCLUSION

Since the inception of the software patent, courts have created an environment in which overly broad, poorly disclosed software patents receive insufficient scrutiny. This leads to software patents that do not serve the fundamental goals of US patent law. The confusion over patentable subject matter is a symptom, not the disease. That problem goes deeper—to the nature of the software industry and the courts’ caricatures of the average programmer’s skill. The combination of broad grants at the USPTO and subsequent invalidations in litigation have not significantly slowed the production of software innovation but have served to silo that information within the firms that are innovating.

Patent law is a strong policy tool for encouraging disclosure for the “promot[ion] . . . of . . . useful [a]rts.”289 By returning to the utilitarian patent-eligibility test of State Street Bank, courts can resolve the confusion over what is, and is not, an abstract idea. By making the determination of the level of skill of a PHOSITA for claim-construction purposes a question of fact, the court can de-calcify the view of programmers as persons of astonishing ability. This, in turn, would reinvigorate the enablement and best-mode requirements of patentability, resulting in more-thorough disclosure and narrower patent claims. Following this blueprint will allow software to take its

285. See id.
287. See supra notes 1–4 and accompanying text.
288. See supra notes 1–4 and accompanying text.
place among the other innovative industries and prevent it from causing perennial legal controversies. Further, those software patents can provide a treasure trove of useful technical data to future software innovators.

* Parker Hancock


* J.D. Candidate 2014, Vanderbilt Law School. The Author would like to thank the Vanderbilt Journal of Entertainment & Technology Law editorial staff for their hard work and valuable input in the publication of this Note. I would also like to thank Professor Sean Seymore for his excellent instruction and input, without which this Note could not have been written.