University of Richmond
UR Scholarship Repository

Law Faculty Publications | School of Law

2014

Debugging Software's Schemas

Kristen Osenga
University of Richmond, kosenga@richmond.edu

Follow this and additional works at: http://scholarship.richmond.edu/law-faculty-publications

 Part of the Intellectual Property Law Commons

Recommended Citation
Kristen Osenga, *Debugging Software's Schemas*, 82 Geo. Wash. L. Rev. 1832 (2014).

This Article is brought to you for free and open access by the School of Law at UR Scholarship Repository. It has been accepted for inclusion in Law Faculty Publications by an authorized administrator of UR Scholarship Repository. For more information, please contact scholarshiprepository@richmond.edu.

Debugging Software's Schemas

*Kristen Osenga**

ABSTRACT

The analytical framework being used to assess the patent eligibility of software and computer-related inventions is fraught with errors, or bugs, in the system. A bug in a schema, or framework, in computer science may cause the system or software to produce unexpected results or shut down altogether. Similarly, errors in the patent eligibility framework are causing unexpected results, as well as calls to shut down patent eligibility for software and computer-related inventions.

There are two general schemas that are shaping current discussions about software and computer-related invention patents—that software patents are generally bad (the bad patent schema) and that software patent holders are problematic (the troll schema). Because these frameworks were created and are maintained through a series of cognitive biases, they suffer from a variety of bugs. A larger flaw in the system, however, is that using these two schemas to frame the issue of patent eligibility for software and computer-related inventions misses the underlying question that is at the heart of the analysis—what is an unpatentable "abstract idea." To improve the present debate about the patent eligibility for these inventions, it is therefore critical that the software patent system be debugged.

TABLE OF CONTENTS

* Professor of Law, University of Richmond School of Law. I would like to thank Michael Risch, Lisa Larrimore Ouellette, Timothy Holbrook, Jim Gibson, and Cynthia Ho for their helpful comments, as well as the members of *The George Washington Law Review* and participants at the *Law Review*'s "Cracking the Code: Ongoing Section 101 Patentability Concerns in Biotechnology and Computer Software" Symposium.

INTRODUCTION

In computer terminology, a schema is a diagram or model used to describe structures for containing and processing data.[1] For example, a database schema may include information about the various fields of the database, the types of data each field may contain, and how the fields may be related.[2] A flawed schema in the computer world potentially results in a bug—an error that results in a computer program or system producing an incorrect result, acting in unexpected ways, or shutting down altogether.[3] In cognitive theory, a schema is a structure or framework that helps organize and interpret information.[4] While cognitive schemas are generally useful because they allow efficient processing of information, they too can lead to incorrect results, unexpected behaviors, or system shutdowns. This erroneous decisionmaking may be due to cognitive biases, such as confirmation bias or stereotyping.[5] These two worlds—computer science and cognitive science—have collided at the intersection of eligibility for patent protection of software and computer-related inventions; unfortunately, the resulting system is in dire need of debugging.[6]

Bugs in the software patent framework are causing problems, largely manifested by a lack of organization and guidance regarding

---

1 *See, e.g.*, *Schema*, TECHTERMS.COM (June 18, 2013), http://www.techterms.com/definition/schema.

2 *See id.*

3 *See, e.g.*, *Bug*, TECHTERMS.COM (Dec. 17, 2008), http://www.techterms.com/definition/bug.

4 *See, e.g.*, MARTHA AUGOUSTINOS, IAIN WALKER & NGAIRE DONAGHUE, SOCIAL COGNITION: AN INTEGRATED INTRODUCTION 68 (2d ed. 2006) (explaining that "[a] schema is conceptualized as a cognitive structure, which contains general expectations and knowledge of the world," and that these structures are used to "select and process incoming information from the social environment"); Ronald Chen & Jon Hanson, *Categorically Biased: The Influence of Knowledge Structures on Law and Legal Theory*, 77 S. CAL. L. REV. 1103, 1131 (2004) ("Categories and schemas are critical building blocks of the human cognitive process. They allow humans to process or at least cope with the infinite amount of information in their environs." (footnote omitted)).

5 *See, e.g.*, Sara Gordon, *Through the Eyes of Jurors: The Use of Schemas in the Application of "Plain-Language" Jury Instructions*, 64 HASTINGS L.J. 643, 657–58 (2013) (noting that confirmation bias occurs when individuals disregard information that contradicts their schemas); *see also* Chen & Hanson, *supra* note 4, at 1231 ("Stereotypes . . . illustrate the difficulty of resisting the potential biases that schemas present.").

6 Debugging is the elimination of errors in computer programs, ideally before releasing the program to the public. *See, e.g.*, *Debug*, TECHTERMS.COM, http://www.techterms.com/definition/debug (last visited Dec. 19, 2014).

the patent eligibility of software and computer-related inventions. Whether, and to what extent, these inventions are eligible for patenting is a complete toss-up under current law, and this lack of certainty is having a widespread effect on the entire patent system.[7] Judicial opinions about software patent eligibility produce unexpected results,[8] legislative proposals attempting to fix such problems might produce incorrect results,[9] and, as some commentators hope, the software patent system is in danger of shutting down altogether.[10]

Legislative and judicial decisionmaking for software patents are influenced by preconceived frameworks. These decisionmakers believe that software patents are generally bad (the bad patent schema) and that software patent holders are problematic (the troll schema).[11] There are two problems with these frameworks. First, the bad patent schema and the troll schema have been created through various cognitive biases, resulting in flaws. Second, these two schemas that are helping to frame the issue for decisionmakers are not the right structures to answer the underlying question about whether patent protection should be available for software and computer-related inventions.[12]

A large number of software patent applications are filed each year,[13] and it is estimated that hundreds of thousands of patents cover-

---

7 *See, e.g.*, Donald S. Chisum, *Weeds and Seeds in the Supreme Court's Business Method Patents Decision: New Directions for Regulating Patent Scope*, 15 LEWIS & CLARK L. REV. 11, 14 (2011) (noting that the software patent framework, as it currently stands, "can lead to subjectively-derived, arbitrary and unpredictable results. This uncertainty does substantial harm to the effective operation of the patent system."); Michael Risch, *Forward to the Past*, 2010 CATO SUP. CT. REV. 333, 362 (noting that *Bilski v. Kappos*, 130 S. Ct. 3218 (2010), "reaffirms decades-old case law—both the substance and the resulting uncertainty"); *see also infra* Part I.B.

8 *See infra* Part II.A.

9 *See, e.g.*, Timothy B. Lee, *Software Patent Reform Just Died in the House, Thanks to IBM and Microsoft*, WASH. POST (Nov. 20, 2013), http://www.washingtonpost.com/blogs/the-switch/wp/2013/11/20/software-patent-reform-just-died-in-the-house-thanks-to-ibm-and-microsoft/ (citing a letter signed by IBM, Microsoft, and others claiming that the reform measures "could harm U.S. innovators by unnecessarily undermining the rights of patent holders" (internal quotation marks omitted)).

10 Some are not shy about their willingness to kill software patents. *See, e.g.*, Colleen V. Chien, *Reforming Software Patents*, 50 HOUS. L. REV. 325, 352 (2012) (noting that abolishing software patents "has enormous popular appeal" as well as "historical and recent precedent"). Others take a more measured approach. *See, e.g.*, JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 235–53 (2008) (allowing for reform of software patents but freely accepting exclusion of software patents if reform is unsuccessful).

11 *See infra* Part II.A.

12 *See infra* Part II.A.

13 Although the United States Patent and Trademark Office ("USPTO") has no classification specifically directed towards software and computer-related inventions, it does try to quan-

ing software and computer-related inventions are in force.[14] Although efficient information processing via schemas and other cognitive biases has its place, there are times when objective, deliberate, and careful consideration of an issue is more appropriate.[15] Whether software and computer-related inventions are patent eligible is far too important a question to rely on biased or incorrect schemas.

Before the conversation goes any further, we should try to debug the software patent schemas. To be sure, it is not possible to fully debug the system; cognitive biases can never be completely eliminated, and some level of shortcut is desirable when assessing the vast number of patent applications filed each year.[16] But with an awareness of these bugs in the software patent system, we should be better able to make a principled, objective decision about the patent eligibility of software and computer-related inventions.[17]

This Essay proceeds in three parts. Part I discusses the current state of patent eligibility for software and computer-related inventions, detailing the incorrect results, unexpected behavior, and system shutdowns caused by the bugs in the system. Part II explains the schemas behind the chaos in the patent system and explains how some relevant cognitive biases are implicated in the creation and maintenance of these frameworks. Part III explains why these schemas, even if not biased, are not the right framework to use in analyzing the question of patent protection for software and computer-related inventions. Although this Essay does not propose a test for patent eligibility of these inventions, or even defend the position that these

---

tify how many "software" patents it issues each year, stating that as many as one-half of the nearly 250,000 patents issued annually are directed towards software inventions. *See* U.S. GOV'T ACCOUNTABILITY OFFICE, GAO-13-465, INTELLECTUAL PROPERTY: ASSESSING FACTORS THAT AFFECT PATENT INFRINGEMENT LITIGATION COULD HELP IMPROVE PATENT QUALITY 12 fig.1 & n.27 (2013).

14 *See, e.g.*, Mark A. Lemley, *Software Patents and the Return of Functional Claiming*, 2013 WIS. L. REV. 905, 928.

15 *See* Cynthia M. Ho, *Drugged Out: How Cognitive Bias Hurts Drug Innovation*, 51 SAN DIEGO L. REV. 419, 436–37 (2014) (noting that quick and intuitive decisionmaking is useful when avoiding a car accident, while at other times, deliberation is a better course of action).

16 In 2012, a total of 542,815 utility patent applications were filed. *See, e.g.*, *U.S. Patent Statistics Chart Calendar Years 1963–2013*, U.S. PAT. & TRADEMARK OFF. (July 24, 2014, 6:22 PM), http://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm.

17 *See* Ian Weinstein, *Don't Believe Everything You Think: Cognitive Bias in Legal Decision Making*, 9 CLINICAL L. REV. 783, 792 (2003); *see also* Troy A. Paredes, *Too Much Pay, Too Much Deference: Behavioral Corporate Finance, CEOs, and Corporate Governance*, 32 FLA. ST. U. L. REV. 673, 739 (2005) ("Making [decisionmakers] aware of their cognitive tendencies and how they process and interpret information (that is, teaching [decisionmakers] how they deviate from perfect rationality) can mitigate cognitive bias.").

inventions should be eligible for patent protection, it presents additional information to the ongoing software patent conversation.

I. THE STATE OF SOFTWARE PATENTS

A discussion of the current state of patent eligibility for software and computer-related inventions naturally must begin with a definition of what is even meant by "software." After defining software, the present state of patent eligibility jurisprudence for these inventions is reviewed.

A. What Is Software

Defining software is no easy task. Given that we are discussing technology, one potential definition would be a technical definition. For example, the Institute of Electrical and Electronics Engineers ("IEEE") defines software as "[c]omputer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system."[18] Another option would be to start with a legal definition for software; after all, we are considering legal rights in the form of a patent. However, the United States Patent and Trademark Office ("USPTO") does not have a specific classification for software patents.[19] Studies on software patents generally define such patents as "a logic algorithm for processing data that is implemented via stored instructions residing on a disk."[20] Using this definition, researchers often identify software patents by two main methods—keyword searches and USPTO technology classes, such as data processing (USPTO technology classes 700–707 and 715–717).[21]

---

18 *See* INST. OF ELEC. & ELECS. ENG'RS, IEEE STANDARD GLOSSARY OF SOFTWARE ENGINEERING TERMINOLOGY 66 (1990).

19 *See US Classes by Number with Title Menu*, U.S. PAT. & TRADEMARK OFF., http://www.uspto.gov/web/patents/classification/selectnumwithtitle.htm (last visited Dec. 5, 2014).

20 James Bessen, *A Generation of Software Patents*, 18 B.U. J. SCI. & TECH. L. 241, 251 (2012); *see also* Sebastian von Engelhardt, *The Economic Properties of Software* (Jena Econ. Research Papers, Paper No. 2008-045, 2008), *available at* http://hdl.handle.net/10419/25729.

21 *See* Bessen, *supra* note 20, at 251–52. Bessen, for example, used USPTO classes for data processing (class numbers 700–707 and 715–717) and other classes that are reliant on software—for example, coded data generation (class number 341), computer graphics processing (class number 345), multiplex communication (class number 370), digital communications (class number 375), cryptography (class number 380), audio signal processing (class number 381), image analysis (class number 382), information security (class number 726), and electronic funds transfer (class number 902). *See id.* at 252 (based on USPTO classifications as of Dec. 28, 2010). There are other methods that have been used to categorize "software patents." For example, Stuart J.H. Graham and David C. Mowery use International Patent Classification ("IPC") classes, subclasses, and groups. *Intellectual Property Protection in the U.S. Software Industry*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY 219, 232 (Wesley M. Cohen & Stephen A. Merrill

One particular difficulty in defining software is that, due to the uncertain state of patent eligibility for software and computer-related inventions, patent attorneys often draft claims to obscure the true nature of the patented invention.[22] To avoid this, one method would be to define software patents as widely inclusive. For example, a recent patent reform bill, popularly known as the SHIELD Act of 2012,[23] defines software as "any process that could be implemented in a computer regardless of whether a computer is specifically mentioned in the patent," as well as "any computer system that is programmed to perform [such] a process."[24] A "computer" is similarly broadly defined as an "electronic, magnetic, optical, electrochemical, or other high-speed data processing device performing logical, arithmetic, or storage functions."[25]

Another difficulty in defining software is that software is an ever-changing target. The shape and format of software keeps evolving as the machines for which it is written also progress—whereas room-sized computers ran early software using shift registers, now surprisingly powerful software can run on a device that fits in your pocket (or smaller).[26] Today, software companies, Internet and social media companies, hardware manufacturers, nonsoftware firms, and even software users develop software.[27] And while there is software qua software, there is also software in your hybrid car that switches from

---

eds., 2003). For a comparison of various methods of defining "software patents," see Anne Layne-Farrar, Defining Software Patents: A Research Field Guide (Feb. 15, 2006) (unpublished manuscript), *available at* http://ssrn.com/abstract=1818025.

22 *See* Chien, *supra* note 10, at 354; Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 9 (2001) (unveiling the "doctrine of the magic words," the practice of drafting software patent claims to appear to cover something else (internal quotation marks omitted)).

23 Saving High-Tech Innovators from Egregious Legal Disputes (SHIELD) Act of 2012, H.R. 6245, 112th Cong. (2012).

24 *Id.* § 2(a).

25 *Id.*

26 *See, e.g.*, Clayton M. Christensen et al., *The Great Disruption*, FOREIGN AFF., Mar./Apr. 2001, at 80, 83–85 (discussing the evolution of computers from 1946 and the room-sized Electronic Numerical Integrator and Computer ("ENIAC") to modern day personal computers). Obviously, in the time since the Christensen article was written, computers have evolved even further. *See, e.g.*, John Markoff, *From Stanford, a Computer to Push Beyond the Boundaries of Silicon*, N.Y. TIMES, Sept. 26, 2013, at B3 (noting that the "shrinking of transistor size over the last half-century has been important because it has significantly lowered the cost of computing, making it possible to build ever more powerful computers that are faster and cheaper, and consume less power with each generation," and highlighting a new advance using nanotechnology that will shrink transistor size even more).

27 *See* Wendy Seltzer, *Software Patents and/or Software Development*, 78 BROOK. L. REV. 929, 947 (2013).

gasoline power to electric power, software in your washing machine that adjusts the wash cycle depending on how dirty the clothes are, software in your cell phone that knows that it is night and turns off the ringer, and so on.[28] Software may not even run on the device in question, but instead function as some form of client-server application.[29] The reality is that software is everywhere.[30]

B. The Software Patent Mess

If defining software is a difficult task, untangling the fiasco that is the law surrounding patent eligibility for software and computer-related inventions is nearly impossible. Neither the U.S. Court of Appeals for the Federal Circuit nor the Supreme Court has provided any solid framework for determining the level of patent protection, if any, available for these inventions. Without guidance, patent eligibility decisions of the Patent Office, as well as the district and appellate courts, are all over the board.[31] It is no wonder, then, that there are calls to

---

28 *See, e.g.*, Peter Fairley, *Software Looks at the Road Ahead to Boost Hybrid-Car Efficiency*, IEEE SPECTRUM (Feb. 3, 2009, 5:00 AM), http://spectrum.ieee.org/computing/software/software-looks-at-the-road-ahead-to-boost-hybridcar-efficiency (discussing control algorithms in hybrid cars that "plan how and when to use stored battery power so as to burn as little gasoline as possible"); Michael Kanellos, *The Sleeping TV, LED Lights and a Washing Machine That Sees Sweat Stains: The Latest from Japan*, GREEN TECH MEDIA (Oct. 6, 2009), http://www.greentechmedia.com/articles/read/the-sleeping-tv-led-lights-and-a-washing-machine-that-sees-sweat-stains-the (touting a washing machine that detects how dirty clothes are); Justin Shillock, *Silent Time Automatically Silences Your Android Phone Based on Time of Day*, LIFEHACKER (Feb. 18, 2011, 2:00 PM), http://lifehacker.com/5764363/silent-time-automatically-silences-your-android-phone-based-on-time-of-day (describing a phone application that allows a person to silence the ringer at certain times).

29 *See* Seltzer, *supra* note 27, at 954 (noting that technology has changed even further, and that now, client-client-and-multiple-servers is more dominant than client-server).

30 *See* Paul Krill, *Microsoft Exec: The World Runs on Software*, INFOWORLD (Apr. 12, 2010), http://infoworld.com/d/developer-world/microsoft-exec-the-world-runs-software-391 ("Everything is powered by software and developers are the ones who make it all happen." (internal quotation marks omitted)).

31 For an example of conflicting court opinions, compare *CLS Bank Int'l v. Alice Corp. Pty. Ltd.*, 685 F.3d 1341, 1355 (Fed. Cir.) (holding method (software) claims to be eligible for patent protection because the computer limitations "play[ed] a significant part in the performance of the invention or . . . the claims [were] limited to a very specific application of the concept"), *vacated*, 717 F.3d 1269 (Fed. Cir. 2012) (en banc), *aff'd*, 134 S. Ct. 2347 (2014), with *Bancorp Servs., L.L.C. v. Sun Life Assurance Co. of Can. (U.S.)*, 687 F.3d 1266 (Fed. Cir. 2012) (denying patent eligibility where computer limitations were found not to be significant). At the agency level, compare SAP Am., Inc. v. Versata Dev. Grp., Inc., No. CBM 2012-00001 (P.T.A.B. Jan. 9, 2013), with Apple Inc. v. Sightsound Techs., LLC, No. CBM 2013-00019 (P.T.A.B. Oct. 8, 2013).

eliminate patent protection altogether for software and computer-related inventions simply to avoid the chaos.[32]

The statute that defines patent-eligible subject matter is deceptively simple; its interpretation at the hands of the courts is anything but simple. Section 101 of the Patent Act permits patenting of "any new and useful process, machine, manufacture, or composition of matter."[33] This provision has long been construed broadly as encompassing "anything under the sun that is made by man,"[34] excluding only "[t]he laws of nature, physical phenomena, and abstract ideas."[35]

The battle line for the patent eligibility of software and computer-related inventions is in the definition of "abstract idea," or more precisely, when an idea is too abstract to warrant patent protection. As Judge Linn of the Federal Circuit has recently stated:

> The abstractness of the "abstract ideas" test to patent eligibility has become a serious problem, leading to great uncertainty and to the devaluing of inventions of practical utility and economic potential. . . . This court has . . . attempted to define "abstract ideas," explaining that "abstract ideas constitute disembodied concepts or truths which are not 'useful' from a practical standpoint standing alone, i.e., they are not 'useful' until reduced to some practical application." More recently, this court explained that the "disqualifying characteristic" of abstractness must exhibit itself "manifestly" "to override the broad statutory categories of patent eligible subject matter." Notwithstanding these well-intentioned efforts . . . the dividing line between inventions that are directed to patent ineligible abstract ideas and those that are not remains elusive. "Put simply, the problem is that no one understands what makes an idea 'abstract.'"[36]

---

32 *See, e.g.*, Joshua D. Sarnoff, *Patent-Eligible Inventions After* Bilski*: History and Theory*, 63 HASTINGS L.J. 53, 106–07 (2011) (calling for categorical eligibility rules as superior to other means of gatekeeping); Brian J. Love, *Why Patentable Subject Matter Matters for Software*, 81 GEO. WASH. L. REV. ARGUENDO 1, 3 (2012), *available at* http://www.gwlr.org/wp-content/uploads/2012/09/Love_Arguendo_81_1.pdf (noting that although eligibility is not the best solution for the software patent problem, it is "the only defensive mechanism left").

33 35 U.S.C. § 101 (2012).

34 Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980) (quoting S. REP. NO. 82-1979, at 5 (1952)).

35 *Id.*

36 *CLS Bank Int'l*, 685 F.3d at 1348–49 (citations omitted). The very "utility and economic potential" of software and computer-related inventions is why this question is so important. *Id.* at 1349. Further, as noted by Judge Rader in *Research Corp. Technologies v. Microsoft Corp.*, 627 F.3d 859 (Fed. Cir. 2010), the whole point of software is to provide an implementation of an idea designed to reach a *commercially valuable end*, which is the exact opposite of abstractness.

The fact that "no one understands what makes an idea 'abstract'"[37] could be related to the historical path patent eligibility jurisprudence has taken. The course leading to the software patent mess is no less of a "murky morass" than the state of the jurisprudence itself.[38]

In the last few years, there has been a discourse between the Federal Circuit and the Supreme Court in an attempt to define an "abstract idea" that renders an invention ineligible for patenting. Despite the flurry of activity in recent years, the path to the present state of affairs dates back to the 1970s and early 1980s when the Supreme Court provided a relatively unworkable standard in a trilogy of cases concerning early software inventions (the "trilogy cases"): *Gottschalk v. Benson*,[39] *Parker v. Flook*,[40] and *Diamond v. Diehr*.[41] The resulting standard, to the extent there was one, was that claims including algorithms were suspected of being "abstract ideas," and that algorithms per se were not eligible for patenting.[42] Courts used, and struggled with, this standard for nearly a quarter century.[43]

Then, in the late 1990s, the Federal Circuit decided a pair of cases widely believed to have opened the doors of the Patent Office to software and business method patents[44]: *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*[45] and *AT&T Corp. v. Excel Communications, Inc.*[46] In these cases, the Federal Circuit implemented the "useful, concrete, and tangible result" test for "abstract ideas"—if an invention produced a useful, concrete, and tangible result, it was

---

See id. at 869 ("[I]nventions with specific applications or improvements to technologies in the marketplace are not likely to be [deemed abstract and unpatentable]."); *see also* Dina Roumiantseva, Note, *The Eye of the Storm: Software and the Abstract Idea Doctrine in* CLS Bank v. Alice, 28 BERKELEY TECH. L.J. 569, 579 (2013).

37 *CLS Bank Int'l*, 685 F.3d at 1349 (internal quotation marks omitted).

38 *See* MySpace, Inc. v. GraphOn Corp., 672 F.3d 1250, 1259–60 (Fed. Cir. 2012).

39 Gottschalk v. Benson, 409 U.S. 63 (1972).

40 Parker v. Flook, 437 U.S. 584 (1978).

41 Diamond v. Diehr, 450 U.S. 175 (1981).

42 *See, e.g.*, Kevin Emerson Collins, *Patent Law's Functionality Malfunction and the Problem of Overbroad, Functional Software Patents*, 90 WASH. U. L. REV. 1399, 1467 (2013).

43 *See id.* at 1467–68 & nn.317–22.

44 *See, e.g.*, Jonathan M. Barnett, *Property as Process: How Innovation Markets Select Innovation Regimes*, 119 YALE L.J. 384, 408 n.55 (2009) (explaining how *AT&T* allowed for patentability of software qua software); *id.* at 415–16 (explaining how the *State Street* decision "explicitly rejected the historical exclusion of business method patents"). Other scholars disagree with this narrative. *See* Risch, *supra* note 7, at 341 (stating that *Diehr* opened the door to software patenting well before *State Street*).

45 State St. Bank & Trust Co. v. Signature Fin. Grp., Inc., 149 F.3d 1368 (Fed. Cir. 1998).

46 AT&T Corp. v. Excel Commc'ns, Inc., 172 F.3d 1352 (Fed. Cir. 1999).

not abstract and could be patented.[47] After some years, the Supreme Court began to signal some discomfort with the viability of the "useful, concrete, and tangible result" test, pointing backwards to the trilogy cases but providing little additional guidance.[48]

After a turn, the en banc Federal Circuit implemented a new test to determine whether an invention was an "abstract idea." This new test, the "machine-or-transformation" test,[49] allowed for patenting of inventions that either: (1) were tied to a "particular machine or apparatus" or (2) transformed an article to a "different state or thing."[50] The Supreme Court immediately took issue with this new test, indicating that it should not be used as the "sole test" for patent eligibility.[51] Instead, the Court again turned back to the trilogy cases, noting that the Court "need not define further what constitutes a patentable 'process,' beyond . . . looking to the guideposts in *Benson*, *Flook*, and *Diehr*."[52] This proved less than helpful.[53]

Lacking a coherent framework or constructive leadership from the Supreme Court, the Federal Circuit continued to flounder, trying to apply vague precedent to today's technology, while navigating the "swamp of verbiage that is § 101"[54] and the Supreme Court's precedent on software patenting. The Federal Circuit also had to contend with increasingly hostile public outcry against patent protection for these types of inventions.[55] Subsequent Supreme Court opinions about patent-eligible subject matter did not provide any additional guidance for defining "abstract ideas."[56] Then, in 2013, the Federal

---

47 *See State Street*, 149 F.3d at 1373; *see also AT&T*, 172 F.3d at 1360–61 (applying *State Street*'s "useful, concrete, and tangible result" test to software-related inventions).

48 *See* Lab. Corp. of Am. Holdings v. Metabolite Labs., Inc., 548 U.S. 124, 134–36 (2006) (Breyer, J., dissenting).

49 *See In re* Bilski, 545 F.3d 943, 955 (Fed. Cir. 2008) (en banc), *aff'd sub nom.* Bilski v. Kappos, 130 S. Ct. 3218 (2010).

50 *Id.* at 954.

51 *See* Bilski v. Kappos, 130 S. Ct. 3218, 3221 (2010).

52 *See id.* at 3222.

53 *See, e.g.*, Collins, *supra* note 42, at 1458 (describing how, in *Bilski*, the Supreme Court "held that its earlier (and difficult to parse) opinions" in the trilogy cases were the ultimate test, and that the Court's holding was based on rhetoric, rather than reasoning, from those cases).

54 MySpace, Inc. v. GraphOn Corp., 672 F.3d 1250, 1259–60 (Fed. Cir. 2012).

55 For one boisterous, although not unrepresentative, example of the public's outcry against software patents, consider Mark Cuban, who established the "Mark Cuban Chair to Eliminate Stupid Patents." Cuban characterizes software patents as "'stupid' patents that should have been completely abolished or at least have a shorter legal life." *See, e.g.*, Efrat Kasznik, *Troll Slayer: Can Mark Cuban Cure the U.S. Patent System?*, VENTUREBEAT (Feb. 9, 2013, 12:59 PM), http://venturebeat.com/2013/02/09/troll-slayer-can-mark-cuban-cure-the-u-s-patent-system/.

56 *See* Ass'n for Molecular Pathology v. Myriad Genetics, Inc., 133 S. Ct. 2107, 2116 (2013)

Circuit published an en banc opinion in *CLS Bank International v. Alice Corp. Pty. Ltd.*[57] In an attempt to provide clarity about patent protection for software and computer-related inventions, the Federal Circuit's decision instead included seven separate opinions, representing at least three distinct viewpoints on the subject.[58] The only thing a majority of the court agreed on was that the invention in question was not eligible for patenting; there was no agreement as to why.[59] At the annual meeting of the American Intellectual Property Association in May 2013, former Chief Judge Rader noted that the failure of the Federal Circuit to provide guidance about software patentability in the *CLS Bank* case was the "greatest failure" in his judicial career.[60]

The Supreme Court granted certiorari in the *CLS Bank* case on December 6, 2013.[61] After hearing arguments on March 31, 2014, the Supreme Court issued its opinion on June 19, 2014.[62] Unfortunately, the Court still provided no guidance on what constitutes an abstract idea, and in fact explicitly dodged the question.[63] Although the Court did not specifically prohibit patenting of this type of invention, the Court's advice consisted generally of reference to the early trilogy cases,[64] leaving inventors, practitioners, and courts to muddle through on the issue of patent eligibility for software and computer-related inventions. Although the *CLS Bank* decision could be seen as software's definitive trip to the Supreme Court, the lack of guidance provided by the Court means that questions will continue to persist. For this reason, addressing the bugs in software's schemas remains a primary issue in patent eligibility jurisprudence going forward.

(discussing what is ineligible as a "naturally occurring phenomena"); Mayo Collaborative Servs. v. Prometheus Labs., Inc., 132 S. Ct. 1289, 1294 (2012) (finding inventions that were "well-understood, routine, conventional activity previously engaged in by researchers in the field" to be ineligible for patenting).

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

58 *See id.*

59 *See id.* at 1273 (per curiam); *see also* Mike Masnick, *Supreme Court to Hear Key Case on Software Patents that Appeals Court Couldn't Figure Out*, TECHDIRT (Dec. 6, 2013, 3:43 PM), http://www.techdirt.com/articles/20131206/15334125492/supreme-court-to-hear-key-case-software-patents-that-appeals-court-couldnt-figure-out.shtml (noting that the Federal Circuit opinion "was one of the biggest judicial messes you'll ever see," with "135 pages of different judges all disagreeing with each other" and "only one single paragraph that the court agreed on").

60 Brian Mahoney, *Software Patent Ruling a Major Judicial Failure, Rader Says*, LAW360 (Oct. 25, 2013, 6:36 PM), http://www.law360.com/articles/482264.

61 Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 734 (2013) (mem.).

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

63 *See id.* at 2357 ("In any event, we need not labor to delimit the precise contours of the 'abstract ideas' category in this case.").

64 *See id.* at 2357–59 (describing *Benson*, *Flook*, and *Diehr*, as well as *Bilski*).

II. THE BIASES IN SOFTWARE'S SCHEMAS

Cognitive science has long studied how people make judgments and choices, often in ways that seem irrational.[65] In part, these poor choices are made because the human brain can only manage so much information, and effective navigation of daily life requires efficient management of scarce cognitive resources.[66] To aid in processing the vast amount of information people face on a daily basis, humans use cognitive biases—various filters and heuristics—as shortcuts, rather than relying on deliberative, deductive logic.[67] The two primary shortcuts are schemas and heuristics.[68] Schemas, as noted previously, are frameworks that help to organize and interpret information, while helping to avoid irrelevant information.[69] Heuristics are mental shortcuts, or "rules of thumb," that permit information to be processed quickly.[70]

For the most part, schemas and heuristics are helpful and necessary. They can, however, lead to poor decisionmaking if there are flaws in the frameworks that distort how the information is processed, or bugs that focus attention on irrelevant data rather than pertinent details.[71] In processing the vast information related to the patent eligibility of software, we have been relying on two schemas, both of which have been created and maintained by additional cognitive biases or shortcuts.

A. *Defining Software's Schemas*

As noted above, the current software patent mess is unlikely to be fixed without debugging the schemas that undergird how information about software and computer-related inventions is understood. This is because schemas "affect our perception of new information," as well as the decisions we make "based on that information."[72] The

65 *See, e.g.*, Adam S. Zimmerman, *Funding Irrationality*, 59 DUKE L.J. 1105, 1108 (2010).

66 *See* Jeffrey J. Rachlinski & Cynthia R. Farina, *Cognitive Psychology and Optimal Government Design*, 87 CORNELL L. REV. 549, 555 (2002); Jeffrey J. Rachlinski, *The Uncertain Psychological Case for Paternalism*, 97 Nw. U. L. REV. 1165, 1170–71 (2003).

67 *See* Derek E. Bambauer, *Shopping Badly: Cognitive Biases, Communications, and the Fallacy of the Marketplace of Ideas*, 77 U. COLO. L. REV. 649, 673 (2006); Peter Lee, *Patent Law and the Two Cultures*, 120 YALE L.J. 2, 22–24 (2010); Rachlinski, *supra* note 66, at 1170–71.

68 *See* Rachlinski & Farina, *supra* note 66, at 555.

69 *See id.* at 555–56.

70 *See id.* at 555.

71 *See id.* at 555–58.

72 *See, e.g.*, Gordon, *supra* note 5, at 652.

current understanding of software patents can be broken into two primary schemas—the bad patent schema and the troll schema.

1. *The Bad Patent Schema*

The bad patent schema begins with the premise that the patent system as a whole is bad,[73] and that software patents are even worse due to a nonsensical standard for patent-eligible subject matter and overly broad and vague claim language.[74] Because bad patents have not been curbed through other mechanisms, software and computer-related inventions should be denied patent protection, generally under the "abstract idea" exception.[75] This is appealing because patent eligibility has often been referred to as a "threshold test"[76] or a "screening" device,[77] allowing for quick disposal of these undesirable patent applications.

There are two main arguments used to support the bad patent schema. These arguments are (1) patents are not required for innovation in the field of software and computer-related inventions and (2) software patents are too broad, poorly examined, and include inadequate disclosure.

First, patent eligibility for software and computer-related inventions is often challenged on the notion that patents are not required as an "incentive to innovate."[78] If software patents should be granted as an incentive, then the questions that should be asked include: what level of innovation would occur without a patent grant, whether a patent grant would cause a greater loss to society than the benefit it provides, and whether a line can be drawn between subject matter that

---

73 Alternatively, it is possible that the patent system as a whole is not bad, but that a lot of bad, or invalid, patents are being issued by the Patent Office. *See, e.g.*, Timothy Holbrook, *Not All Patent Trolls Are Demons*, CNNOPINION (Feb. 21, 2014, 9:08 AM), http://www.cnn.com/2014/02/21/opinion/holbrook-patent-trolls-demons/index.html.

74 *See, e.g.*, Julie Samuels, *Finally: This Is How to Fix the 'Patent Fix' We're All In*, WIRED (Apr. 2, 2013, 9:30 AM), http://www.wired.com/opinion/2013/04/this-is-how-to-fix-the-patent-fix-were-in/.

75 *See, e.g.*, BESSEN & MEURER, *supra* note 10, at 235–53; Sarnoff, *supra* note 32, at 106–07; Love, *supra* note 32, at 2–3.

76 *See* Bilski v. Kappos, 130 S. Ct. 3218, 3225 (2010).

77 *See* Mayo Collaborative Servs. v. Prometheus Labs., Inc., 132 S. Ct. 1289, 1303 (2012); *see also In re* Comiskey, 554 F.3d 967, 973 (Fed. Cir. 2009) ("Only if the requirements of § 101 are satisfied is the inventor 'allowed to pass through to' the other requirements for patentability, such as novelty under § 102 and . . . non-obviousness under § 103." (quoting *In Re* Bergy, 596 F.2d 952, 960 (C.C.P.A. 1979))).

78 *See, e.g.*, David S. Olson, *Taking the Utilitarian Basis for Patent Law Seriously: The Case for Restricting Patentable Subject Matter*, 82 TEMP. L. REV. 181, 183–84 (2009); Seltzer, *supra* note 27, at 929.

needs protection and subject matter that does not.[79] In answering these questions, a common response is that, even if an incentive is needed, software patents provide very little societal benefit because of inadequate disclosure.[80] Specifically, the application can describe what the software should do, in sufficiently specific terms to obtain a patent, without providing any details about how the software will actually work.[81] What is being patented is not an invention, but rather simply an unimplemented idea.

Some commentators argue that even an ideal software patent—with complete disclosure and covering a completed invention—does not deserve patent protection, because it does not induce the development of innovative software, as the monopoly period is unnecessary to recapture development investments.[82] The nature of software development does not require patent incentives, because there is a low bar to entry, capital costs are low, and human capital requirements are small; even a single programmer can make significant progress on a project.[83] Although software development is uncertain, software development permits "rapid prototyping" and the ability to "release early, release often"—with bug fixes being available even after a product release.[84]

Second, many commentators complain that software patents are too broad, poorly examined, and inadequately disclosed,[85] even where the underlying invention may be more than a simple unimplemented idea. Software patents have "notoriously fuzzy" boundaries, making it difficult to determine where the rights of the inventor end and public domain begins.[86] The fuzzy boundaries also make examination difficult, as it is not easy to tell when the software patent application overlaps prior art.[87] Another problem is that prior art in the software and computer-related inventions field is allegedly difficult to find, leaving examiners without the resources to reject claims that are not

79 *See* Olson, *supra* note 78, at 184.

80 *See, e.g.*, Greg R. Vetter, *Patent Law's Unpredictability Doctrine and the Software Arts*, 76 MO. L. REV. 763, 776 (2011).

81 *See* Seltzer, *supra* note 27, at 944.

82 *See id.* at 930, 943.

83 *See id.* at 944, 975; Samuels, *supra* note 74.

84 *See* Seltzer, *supra* note 27, at 956–57.

85 *See, e.g.*, Collins, *supra* note 42, at 1400.

86 Peter Menell, *It's Time to Make Vague Software Patents More Clear*, WIRED (Feb. 7, 2013, 4:10 PM), http://www.wired.com/opinion/2013/02/its-time-to-make-vague-software-patents-more-clear/.

87 *See* Seltzer, *supra* note 27, at 955.

novel or nonobvious.[88] Finally, to the extent these inventions do not require patent protection in order to incentivize invention, the multitude of patent applications pending at the Patent Office is simply an extra burden,[89] further exacerbating examination difficulties and the issuance of bad patents.[90]

2. *The Troll Schema*

The other framework manipulating our understanding of software patents is the troll schema. The argument here is that the "troll," or patent assertion entity ("PAE"), problem is particularly present in the software industry.[91] Commentators justify the troll schema on the basis that the software industry has a low barrier to entry, and software patent quality is suspect, so it is easy to obtain these patents (either ab initio or from a failed company).[92] These patents are then asserted, at times relentlessly, against successful producers of products that incorporate the software.[93] Troll litigation has also been labeled "simple extortion" and even "a 'Tony Soprano' protection racket."[94] Because software patents are nearly five times as likely to be litigated as other patents, they are more susceptible to abuse by trolls.[95]

---

88 *See* Cohen & Lemley, *supra* note 21, at 42; James Gleick, *Patently Absurd*, N.Y. TIMES MAG., Mar. 12, 2000, at 50, *available at* http://www.nytimes.com/2000/03/12/magazine/patently-absurd.html. One problem with this argument, however, is that it is getting stale. Because software and business method inventions have been widely considered to be eligible for patenting since *State Street* in 1999, the amount and availability of prior art must have increased. Yet current articles relying on the bad patent schema still rely on prior art arguments from over a decade ago. Consider Wendy Seltzer's article from 2013. Seltzer, *supra* note 27. For the proposition that the USPTO has inadequate information to properly examine software-type patent applications, Seltzer cites an article from 2000. *See id.* at 954–55 & n.129 (citing Richard S. Gruner, *Better Living Through Software: Promoting Information Processing Advances Through Patent Incentives*, 74 ST. JOHN'S L. REV. 977, 1063–64 (2000)).

89 *See* BESSEN & MEURER, *supra* note 10, at 247.

90 *See* Olson, *supra* note 78, at 189.

91 *See, e.g.*, Lemley, *supra* note 14, at 932 ("Patent 'trolls' . . . are legion in the software industry.").

92 *See, e.g.*, Seltzer, *supra* note 27, at 976–77.

93 *See id.*

94 *See* N.V., *Obituary for Software Patents*, ECONOMIST (Dec. 13, 2013, 6:32 AM), http://www.economist.com/blogs/babbage/2013/12/difference-engine-0 (internal quotation marks omitted).

95 *See* Samuels, *supra* note 74.

B. *Additional Cognitive Biases*

Schemas are one type of cognitive bias,[96] but there are additional cognitive biases that are particularly relevant in the development and maintenance of the flawed schemas that influence the patent eligibility analysis for software and computer-related inventions.[97] These include confirmation bias, availability bias, and grouping biases.[98]

1. *Confirmation Bias*

Confirmation bias is the natural tendency to reinforce beliefs by seeking out consistent information, ignoring inconsistent information, and, when faced with consistent and inconsistent information, giving greater weight to evidence that validates the existing belief.[99] Typically, confirmation bias is less likely to occur when the cost of making incorrect decisions is high.[100] When faced with conflicting information, we prefer information that supports our perspective, and are unlikely to shift our conclusions simply because we receive additional or better information.[101] We also have great difficulty when processing adverse information that is posed in the negative or asymmetrically.[102]

For an example of how confirmation bias may be at play in the bad patent schema, consider the empirical work of Professors John Allison and Ronald Mann.[103] In particular, they performed a study of the quality of software and nonsoftware patents, looking at "(1) the number of claims in the patent, (2) the number of prior art references in the patent, and (3) the number of forward citations to the patent."[104] Although they admit their work is suggestive due to the com-

---

96 *See, e.g.*, Gregory S. Alexander, *A Cognitive Theory of Fiduciary Relationships*, 85 CORNELL L. REV. 767, 768–69 (2000).

97 For a broader claim, see Lisa Larrimore Ouellette, *Cultural Cognition of Patents*, 4 IP THEORY 28, 33 (2014) (claiming that "cultural cognition likely contributes to the dysfunctional public discourse over patents").

98 *See infra* Part II.B.1–3.

99 *See* Margit E. Oswald & Stefan Grosjean, *Confirmation Bias*, *in* COGNITIVE ILLUSIONS: A HANDBOOK ON FALLACIES AND BIASES IN THINKING, JUDGMENT AND MEMORY 79 (Rüdiger F. Pohl ed., 2004); *see also, e.g.*, SCOTT PLOUS, THE PSYCHOLOGY OF JUDGMENT AND DECISION MAKING 233 (1993); ARTHUR S. REBER, THE PENGUIN DICTIONARY OF PSYCHOLOGY 151 (2d ed. 1995); Michael A. McCann, *It's Not About the Money: The Role of Preferences, Cognitive Biases, and Heuristics Among Professional Athletes*, 71 BROOK. L. REV. 1459, 1460 (2006).

100 *See* Oswald & Grosjean, *supra* note 99, at 91–92.

101 *See id.*

102 *See* Bambauer, *supra* note 67, at 679.

103 *See* John R. Allison & Ronald J. Mann, *The Disputed Quality of Software Patents*, 85 WASH. U. L. REV. 297 (2007).

104 *See id.* at 321. These factors were chosen because of their dominance in existing empirical literature. *See id.*

plexity of the questions at issue, they found that "by objective standards, software patents as a group compare quite favorably to patents that the same firms are obtaining, at the same time, on non-software inventions."[105] Further, they found that "patents obtained by small firms are no worse than the patents of the large firms."[106]

The findings in this study provide some evidence that is contrary to both the bad patent schema and the troll schema. In their article, Allison and Mann specifically take issue with the bad patent schema, stating that their findings "undercut the common suggestions that software patents should be prohibited entirely or should face special hurdles for examination designed to stem the alleged flood of low-quality patents."[107] Although the article does not particularly address the troll schema, patents held by trolls generally come from small firms;[108] Allison and Mann's findings suggest that patents from small firms are no worse in quality than patents obtained by large firms.[109]

Despite Allison and Mann's work, confirmation bias may make it difficult for other scholars to accept this information, because it is inconsistent with the popularly held bad patent schema and troll schema. Confirmation bias makes scholars prefer information that supports their own perspectives.[110] Consider the following: a quick legal research search for law review articles about bad patents or patent quality yielded over 200 results.[111] However, only fifteen articles cited Allison and Mann's study.[112] Of these fifteen articles, only one explicitly cited the study's findings with approval.[113] On the other hand, at

105 *See id.* at 333–34.

106 *See id.* at 334.

107 *See id.* In the same vein, there is more recent research that suggests that software patent applications are examined at least as rigorously as nonsoftware patent applications. *See* Stuart Graham & Saurabh Vishnubhakat, *Of Smart Phone Wars and Software Patents*, 27 J. ECON. PERSPS. 67, 73 (2013).

108 *See* Michael Risch, *Patent Troll Myths*, 42 SETON HALL L. REV. 457, 486–87 (2012).

109 *See* Allison & Mann, *supra* note 103, at 334.

110 *See* Christopher R. Leslie, *Rationality Analysis in Antitrust*, 158 U. PA. L. REV. 261, 314 (2010).

111 A search on Lexis Advance consisting of "software /p patent /p quality" in the Secondary Materials: Law Reviews and Journals database from 1/1/2007 through 12/31/2013 yielded 230 results, including the Allison and Mann article. A similar search consisting of "software /p patent /p bad" yielded 167 results, in addition to the Allison and Mann article.

112 This data is a result of running a Shepard's search in Lexis Advance. The search revealed that fifteen law review articles cited the article. To be sure, there are other descriptive empirical studies that may provide similar information; however, this is exemplary. Future research would be useful to consider other confirmation bias arguments that may persist concerning the troll schema.

113 *See* Barnett, *supra* note 44, at 428 n.103. Other citing articles refer to Allison and Mann's definition of "software." *See, e.g.*, Stephen Clowney, *Property in Law: Government*

least five cited the article as a contrary, or conflicting, data point.[114] The other articles about software patent quality that followed the Allison and Mann article failed to even mention the study's opposing data.[115] To be sure, there are many reasons behind why scholars cite and fail to cite other scholarship; however, one inference is that confirmation bias is affecting our intake of new or additional information about software's schemas, which results in authors choosing to avoid referencing the Allison and Mann article.

2. *Availability Bias*

Availability bias explains how the amount and source of information affects decisionmaking. This bias reflects that people assess frequency or probability based on the ease with which information about an event can be recalled.[116] Because it is easier to bring to mind a vivid or sensational story or a story that receives a large amount of media attention, rather than routine stories of everyday activity, we are more likely to overestimate the presence of the sensational activity.[117] Similarly, the availability cascade refers to the old adage that if you repeat something often enough, it will become true. More formally, scholars have defined the availability cascade as "a self-reinforcing process of collective belief formation by which an expressed perception triggers a chain reaction that gives the perception increasing plausibility through its rising availability in public discourse."[118] The effect of availability bias can be heightened if individuals lack suf-

---

Rights in Legal Innovations, 72 OHIO ST. L.J. 1, 46 n.193 (2011) (referring the reader to Allison and Mann's definition of software patents).

114 *See, e.g.*, Bernard Chao, *Finding the Point of Novelty in Software Patents*, 28 BERKELEY TECH. L.J. 1217, 1224 n.48 (2013) (citing Allison and Mann's article with a "but see" signal following the author's statement that software patents are of low quality); Tun-Jen Chiang, *The Rules and Standards of Patentable Subject Matter*, 2010 WIS. L. REV. 1353, 1407 n.266 (using Allison and Mann as a comparison to another study to illustrate conflicting views on patent quality); Jeanne C. Fromer, *The Compatibility of Patent Law and the Internet*, 78 FORDHAM L. REV. 2783, 2796 n.88 (2010) ("*That said*, Allison and Mann demonstrate . . . that software patents are indistinguishable [from others patents]." (emphasis added)); Stephen McJohn, *Scary Patents*, 7 NW J. TECH. & INTELL. PROP. 343, 344 n.12 (2009) (citing Allison & Mann's article with a "but see" signal); Vetter, *supra* note 80, at 776 & n.52 (referring to Allison and Mann as "contrarians among the commentators").

115 *See, e.g.*, Alan Devlin, *Systemic Bias in Patent Law*, 61 DEPAUL L. REV. 57, 81 n.179 (2011) (referring to the definition of software patents put forth in the Allison and Mann article without discussing the study's results).

116 *See* Amitai Aviram, *The Placebo Effect of Law: Law's Role in Manipulating Perceptions*, 75 GEO. WASH. L. REV. 54, 71–72 (2006).

117 *See id.* at 72.

118 Timur Kuran & Cass R. Sunstein, *Availability Cascades and Risk Regulation*, 51 STAN. L. REV. 683, 683 (1999).

ficient information to form their own beliefs about an issue, or if they adopt the popular, repeated viewpoint to garner approval or simply because other people have also adopted that view (i.e., jumping on the bandwagon).[119]

One example of availability bias in the context of software's schemas is the fact that, although there are some hundreds of thousands of patents on software and computer-related inventions estimated to be currently in force,[120] the most readily available information is focused on the sensational—or more accurately, the underwhelming—patents.[121] The popular media features reports on "notorious" patents, like Amazon's patent for one-click shopping and Priceline's patent for a reverse auction,[122] or other seemingly silly inventions like Apple's patent application for offering author autographs on e-books.[123] Despite the abundance of articles decrying poor software patents, there are few, if any, that highlight the positive patents, like the hybrid engine patent.[124]

The troll schema is similarly reinforced by stories such as the one that appeared in the *Palm Beach Post*, reporting the efforts of patent owner ArrivalStar to enforce its patents.[125] The article noted that ArrivalStar's head had been called not just a "patent troll" but also a "shakedown artist" and a "cockroach."[126] The story upped the ante by reporting that the company was once based in Delray Beach, but has changed incorporation to "the tax haven of Luxembourg."[127] With stories like this, it is hard to recall the many colorful stories written about other patent lawsuits where the invention in question was an

---

119 *See id.* at 685–87.

120 *See* Lemley, *supra* note 14, at 928.

121 This adjective is based on the attitude of the articles, not the author. *See, e.g.*, Tim Cushing, *US Patent Office Grants "Photography Against a White Background" Patent to Amazon*, TECHDIRT (May 8, 2014, 5:41 AM), https://www.techdirt.com/articles/20140507/04102327144/us-patent-office-grants-photography-against-white-background-patent-to-amazon.shtml.

122 *See, e.g.*, Rod Cooper et al., *Patents Are Not the Enemy*, CHI. TRIB., Aug. 15, 2012, at C21 (deeming Amazon's one-click patent "notorious"); Jube Shiver, Jr., *Little Gain Seen in Patent Filings*, L.A. TIMES, Oct. 21, 2002, at C4.

123 *See* L. Gordon Crovitz, Op-Ed., *Information Age: Jimmy Carter's Costly Patent Mistake*, WALL ST. J., Dec. 16, 2013, at A13.

124 Even the articles discussing positive patents have a negative spin. *See, e.g.*, John Murphy, *Toyota Builds Thicket of Patents Around Hybrid to Block Competitors*, WALL ST. J. (July 1, 2009, 11:59 PM), http://online.wsj.com/news/articles/SB124640553503576637.

125 Jeff Ostrowski, *Patent Trolls Build Piles of Cases in Court*, PALM BEACH POST, Sept. 1, 2013, at D1.

126 *Id.*

127 *Id.* Among other irrelevant yet salacious facts, the story also highlighted that the head of ArrivalStar had moved to Canada. *Id.*

important software invention. Or perhaps it is difficult to recall those stories because they have not been written. In any case, the troll schema has, in part, been created, and certainly perpetuated, through accounts such as this.

One final aspect of the availability bias is that information is also recalled based on the perceived importance of its source. When the President of the United States says that there is a patent troll problem, as President Obama did in June 2013,[128] it certainly brings added attention to the schema. This increases the amount of press the framework receives, heightening its credibility—particularly among those who lack information to make an independent decision.[129]

3. *Grouping Biases*

The final set of cognitive biases relates to attributing certain characteristics to a group without regard to individual differences of members of that group. This is often called the stereotyping bias. In these cases, a specific, vivid case will often "evoke affective reactions toward the entire class of objects it represents, despite countervailing but pallid assurances about typicality."[130] The representativeness bias prompts a belief that individuals with one characteristic share a second characteristic, based on how often or how closely individuals with the second characteristic exhibit the first.[131] Although these heuristics are helpful, as individual information is often difficult to ascertain, they can lead people to ignore relevant, actual data.[132]

The problem for software's schemas is that there is no such thing as a typical software or computer-related invention, nor is there one type of patent troll, at least as the term has been broadly applied.[133]

---

128 *See* Edward Wyatt, *Obama Orders Regulators to Root Out "Patent Trolls,"* N.Y. TIMES, June 5, 2013, at B1.

129 Of course, it may not be irrational to trust a public official; however, trusting a public official simply because of his position, rather than any particular expertise, would be an example of placing perceived importance over full information.

130 Richard E. Nisbett et al., *Improving Inductive Inference*, *in* JUDGMENT UNDER UNCERTAINTY: HEURISTICS AND BIASES 445, 454 (Daniel Kahneman et al. eds., 1982).

131 *See* Nancy Levit, *Confronting Conventional Thinking: The Heuristics Problem in Feminist Legal Theory*, 28 CARDOZO L. REV. 391, 396–97 (2006); James S. Liebman et al., *The Evidence of Things Not Seen: Non-Matches as Evidence of Innocence*, 98 IOWA L. REV. 577, 624 (2013).

132 *See, e.g.*, Chris Guthrie et al., *Inside the Judicial Mind*, 86 CORNELL L. REV. 777, 805 (2001).

133 For example, consider the working definition the Federal Trade Commission is using to study patent assertion entities ("PAE") or patent trolls: "PAEs are firms with a business model based primarily on purchasing patents and then attempting to generate revenue by asserting the intellectual property against persons who are already practicing the patented technology" but

As noted above, there is no such thing as a software industry, because software is anywhere and everywhere, produced by multitudes of different producers for different devices and different purposes.[134] Additionally, at least one commentator has identified three types of patent trolls; there may be more.[135] Yet commentary and legislation regarding software or patent trolls generally fail to differentiate at all. Instead, the negative characteristics of one piece of software or one patent holder are being ascribed to all members of the category.[136] For example, although there may be patent trolls that engage in abusive patent litigation tactics, the SHIELD Act of 2013[137] proposes fee-shifting against a losing patent holder in patent infringement cases *unless* the patent holder is the original inventor or assignee, the patent holder can show "substantial investment made . . . in the exploitation of the patent through production or sale of an item covered by the patent," or the patent holder is a university or "technology transfer organization."[138] As another example, consider the breadth of the definition of software in the SHIELD Act of 2012, where fee-shifting would be permitted in the case of litigation involving software patents.[139]

Stereotyping or representativeness bias disposes us to focus on the fact that a software patent may be more likely to be bad or invalid. More importantly, this bias may make us forget that any particular software patent may be just fine, or that enforcement of a software patent, even by an entity that does not produce its own goods, is a valid use of the patent system. Instead, we rely on resemblance and ignore individualized information.

---

not entities "that primarily seek to develop and transfer technology, such as universities, research entities, and design firms." *See* Agency Information Collection Activities, 78 Fed. Reg. 61,352, 61,352 n.1 (proposed Oct. 3, 2013).

134 *See supra* Part I.A.

135 *See* Mark A. Lemley & A. Douglas Melamed, *Missing the Forest for the Trolls*, 113 COLUM. L. REV. 2117, 2126 (2013) (defining the three types of trolls as (1) "a company that owns a patent and hopes to strike it big in court," (2) a company "interested in quick, low-value settlements for a variety of patents" that do not want to actually go to trial, and (3) a company engaged in "patent aggregation"). Another study identifies twelve classes of trolls or nonpracticing entities. *See* John R. Allison et al., *Extreme Values or Trolls on Top? The Characteristics of the Most-Litigated Patents*, 158 U. PA. L. REV. 1, 10 tbl.1 (2009).

136 *But see* Graham & Vishnubhakat, *supra* note 107, at 69 (arguing against the bad patent schema for software patent applications).

137 Saving High-Tech Innovators from Egregious Legal Disputes (SHIELD) Act of 2013, H.R. 845, 113th Cong. (2013).

138 *See id.* § 2(d).

139 *See supra* note 24 and accompanying text.

One objection is that these schemas are not irrational or wrong. After all, it is difficult to find concrete evidence of socially beneficial patents that would not have been invented but for the availability of patent protection. The lack of positive patent troll narratives may very well be due to the fact that there are none. And even if software patents are good for society, perhaps the administrative costs and potential for erroneous patent grants outweigh the benefits. However, even if we cannot prove these schemas to be wrong, we also do not know that they are correct and rational. For this reason, it is inappropriate to use these schemas to undergird the software patent conversation.

C. Debugging the Biases

One way to debug software's schemas is to acknowledge the role of cognitive biases in the bad patent schema and the troll schema, and to acknowledge that these schemas influence decisionmaking by courts[140] and legislators[141] regarding the patent eligibility of software and computer-related inventions. These cognitive biases operate at an unconscious level and are so ingrained that even without our knowledge they may impact discussion and decisionmaking.[142] However, there is research that suggests that our decisionmaking can be improved if we are aware of cognitive biases and consider them while making decisions and seeking solutions.[143]

Consider some popular suggestions to fix the software patent system. One often proposed fix is to rely more heavily on other sections of the Patent Act, such as the novelty, nonobviousness, and written description requirements.[144] This proposal conforms to the bad patent schema, but is perhaps immune to the grouping biases, since it sug-

---

140 *See* RICHARD A. POSNER, HOW JUDGES THINK 68–70 (2008).

141 *See supra* notes 8–10 and accompanying text.

142 *See* Anthony G. Greenwald & Linda Hamilton Krieger, *Implicit Bias: Scientific Foundations*, 94 CALIF. L. REV. 945, 946 (2006).

143 Weinstein, *supra* note 17, at 792; *see also* Paredes, *supra* note 17, at 739.

144 *See, e.g.*, Dennis Crouch & Robert P. Merges, *Operating Efficiently Post-*Bilski *by Ordering Patent Doctrine Decision-Making*, 25 BERKELEY TECH. L.J. 1673, 1674 (2010) (arguing that other sections of the Patent Act should be used to examine patents first, resorting to patent eligibility as a last resort); Kristen Osenga, *Ants, Elephant Guns, and Statutory Subject Matter*, 39 ARIZ. ST. L.J. 1087 (2007) (arguing that patent eligibility is a proxy for other, more proper patentability inquiries); Michael Risch, *Everything Is Patentable*, 75 TENN. L. REV. 591 (2008) (arguing that patent eligibility should not be part of the patentability inquiry). These arguments have been criticized as ineffective or unduly costly. *See* Lemley, *supra* note 14, at 938–39 (arguing that "beefing up examination" to fix bad software patents is too costly); Love, *supra* note 32, at 7 (stating that "sections 102, 103, and 112 have proven woefully ineffective at screening overbroad software patents").

gests increased individualized attention for software patent applications.

Other commentators have proposed a different tactic for fixing the software patent system: adjusting the patent maintenance fee system.[145] For example, one proposal from James Bessen suggests that patent maintenance fees should be greatly increased to compensate for patents' burden on society, and would raise the fees to reflect the likelihood of assertion, which would put an additional tax on software and business methods, as these inventions have been shown to be disproportionately asserted.[146] Alternatively, another proposal asserts that because patent trolls often bring suits late in the life of a patent, maintenance fees could be structured to increase as a patent ages and to include additional late-term fees.[147] These plans do not differentiate among individualized patents or patent holders, and thus may be subject to the grouping biases. In addition, the Bessen proposal analogizes software patents to pollution;[148] this is the type of vivid analogy that may reinforce the availability bias.

For another example, consider the SHIELD Act of 2013. When the Act was introduced in March 2013, Representative Peter DeFazio, the bill's sponsor, asserted that "patent troll suits cost American technology companies over $29 billion in 2011 alone."[149] This figure is the take-home (and oft repeated) message from a study by James Bessen and Mike Meurer.[150] However, a study from another set of well-respected scholars, David Schwartz and Jay Kesan, casts doubt on the

145 *See* Eric Goldman, *Fixing Software Patents* 1, 9 (Santa Clara Univ. Sch. of Law Legal Studies Research Papers Series, Paper No. 01-13, 2013), *available at* http://ssrn.com/abstract=2199180 (summarizing two separate maintenance fee proposals).

146 *See* James Bessen, *Can New Fees Fix the Patent System? Experts Weigh In: Make the Polluters Pay!*, WIRED (Sept. 6, 2012, 2:10 PM), http://www.wired.com/opinion/2012/09/can-new-fees-fix-the-patent-system.

147 *See* Brian J. Love, *An Empirical Study of Patent Litigation Timing: Could a Patent Term Reduction Decimate Trolls Without Harming Innovators?*, 161 U. PA. L. REV 1309 (2013); *see also* Brian J. Love, *Let's Use Patent Fees to Stop the Trolls*, WIRED (Dec. 20, 2012, 3:30 PM), http://www.wired.com/opinion/2012/12/how-to-stop-patent-trolls-lets-use-fees.

148 *See* Goldman, *supra* note 145, at 9 (discussing Bessen's proposal).

149 *See, e.g.*, Adam Mossoff, *The SHIELD Act: When Bad Economic Studies Make Bad Laws*, TRUTH ON THE MARKET (Mar. 15, 2013), http://truthonthemarket.com/2013/03/15/the-shield-act-when-bad-studies-make-bad-laws/.

150 *See* James Bessen & Michael J. Meurer, *The Direct Costs from NPE Disputes*, 99 CORNELL L. REV. 387, 389 (2014).

$29 billion figure.[151] This situation may present another good example of confirmation bias.[152]

Finally, consider the ultimate subject matter related proposal: to ban software patents altogether.[153] The reasons most often given for these proposals are that no software patent can be appropriately examined—grouping bias—and that all software patents and patent applications are drawn to abstract ideas—again, grouping bias.[154] These articles do not consider individual patent applications to determine whether one could be appropriately examined, or is drawn to a non-abstract idea.[155] Additionally, these proposals often appeal to the availability bias, with one such essay starting off with the colorful line that "[i]t's not hard to see why many think software patents are a scourge."[156]

The presence of these schemas and biases in the patent-eligible subject matter debate is skewing the conversation and affecting decisionmaking about how to handle software and computer-related inventions. It may be possible to fix the software patent mess by acknowledging the biases in the bad patent schema and the troll schema. However, as computer programmers know, the process of debugging often introduces new, unintended bugs, or uncovers previously undiscovered errors.[157] What if the errors are not related to the biases in the current schemas at all, but rather arise because we have set up the wrong structure entirely to assess the patent eligibility of software and computer-related inventions? After clearing out the biases, it seems that there is a more critical bug.

---

151 *See* Mossoff, *supra* note 149 (discussing David L. Schwartz & Jay P. Kesan, *Analyzing the Role of Non-Practicing Entities in the Patent System*, 99 CORNELL L. REV. 425 (2014)).

152 *See* Joff Wild, *The PR Genius of Messrs Bessen and Meurer*, IAM MAG. (June 28, 2012), http://www.iam-magazine.com/blog/detail.aspx?g=e780e3b8-715d-484f-9318-d04d81e0e9d8&c=5850974 ("What [is] truly fascinating about the work that Bessen and Meurer do is the extraordinary coverage it gets. Whether it is accurate or not, what it does do very quickly is become an accepted truth in the general media.").

153 *See* Love, *supra* note 32, at 3; Andrew Nieh, Note, *Software Wars: The Patent Menace*, 55 N.Y.L. SCH. L. REV. 295, 299 (2010/11) (proposing a per se exception, barring all software from patent eligibility).

154 *See* Nieh, *supra* note 153, at 299.

155 *See* Chien, *supra* note 10.

156 *See* Love, *supra* note 32, at 1.

157 *See Debug*, TECHTERMS.COM, http://www.techterms.com/definition/debug (last visited Dec. 19, 2014).

III. A More Critical Bug in the Schemas

Schemas, whether cognitive or computer-related, are supposed to help organize and process relevant information and data. Yet the schemas that are influencing the software patent analysis are of questionable relevance, at least as the question is currently framed. We are trying to define an "abstract idea," but instead we are asking whether a patent is necessary for this type of invention or whether some holders of these patents engage in abusive litigation behavior. Only one part of the current bad patent schema is even marginally related to the "abstract idea" question—and that deals with the potentially overbroad and vague claims that some software patents include.[158]

Consider an analogy from the computer schema field. There is a database that includes vital information about a person. This database may include fields for the person's name, place of birth, parents, and so on. The question at issue is whether or not the person is of legal driving age. It would be logical to query the database for an "age" field as a direct inquiry and then compare the results from that field to the legal driving age. Or, if no "age" field were available, then perhaps the system would query the database for a "birthdate" field (permitting an analytical or computational inquiry based on the results of that field). Imagine, though, that in trying to determine whether the person was of legal driving age, the system was built to query the "gender" field or the "parents' nationality" field. Clearly, it would be difficult to determine if the person were of legal driving age if the system's response was "female" or "Norwegian."

This is very similar to what is happening in the current conversations about patent eligibility for software and computer-related inventions. Rather than trying to answer the question of whether the particular invention is an ineligible "abstract idea" by direct or analytical inquiry, we are instead asking whether it is software (gender) or who the patent holder is (parents' nationality). Although the question of whether the invention is software may have some relevance to the question of abstraction, the nature of the patent holder is completely immaterial.[159] It is unlikely that the conversation about patent eligibility for software and computer-related inventions will reach the critical question of "abstract idea" as long as the analysis is obscured by irrelevant data. The failure to ask the right question has substantial conse-

158 *See supra* Part II.A.1.

159 To the extent that anything needs to be done about the patent troll problem, it is an issue separate and apart from software and has no relation to the definition of "abstract idea."

quences, because not all software is undeserving of patent protection.[160]

CONCLUSION

This Essay does not pretend to solve the perplexing problem of patent eligibility for software and computer-related inventions. Rather, its purpose is to add some awareness to the underlying schemas and shortcuts that are influencing judicial, legislative, and public perceptions about these inventions. Starting from a position of cognizance should result in a better conversation—less one-sided and more deliberative and objective—going forward. Further, we should be aware that the question we ultimately want to answer, defining "abstract idea," may require us to shift the framework away from easy inquiries and instead dig deeper for a workable analysis. Whether and to what extent software and computer-related inventions should be eligible for patenting is not a question that should be answered on intuition, based on selective information or overly simplistic proxies; it should receive its due consideration, and only then should we be comfortable in releasing the newly debugged software patent schemas to the public.

160 *See* DAN L. BURK & MARK A. LEMLEY, THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT 157–58 (2009) (noting that there are software inventions that deserve protection); Note, *Everlasting Software*, 125 HARV. L. REV. 1454, 1475 (2012) ("Cutting back on the software patent regime risks cutting back on many innovative, good patents in addition to the potentially bad ones.").