THE JOHN MARSHALL
REVIEW OF INTELLECTUAL PROPERTY LAW

OPEN SOURCE PARADIGM: BEYOND THE SOLUTION TO THE SOFTWARE PATENTABILITY DEBATE

GIOVANNA MASSAROTTO

ABSTRACT

Around 300 BCE, a Greek mathematician, Euclid discovered a theorem on which modern geometry and a fundamental algorithm is based. Euclid's theorem represents a method for calculating the greatest common divisors between two integers. Since 300 BCE, both Euclid's Theorem and algorithm have been applied in many fields, including algebra and geometry. But what would have happened if Euclid's Theorem had been patented? The issue is not whether we can continue to use Euclid's Theorem without paying royalties, but if software and algorithms underlying the software are patentable. Although software is based on algorithms similar to the algorithm discovered by Euclid, in the United States software is generally patented.

Open source paradigm, explored here, could resolve the software patentability issue, in addition to representing the best economic/social paradigm to apply in the technology industry. Android, Google's open source operating system, for example, is installed in more than eighty percent of worldwide smartphones, showing that open source strategy is workable. The Web, developed by Tim Berners-Lee, is another of these examples. If the Web had been patented, innovation and the development of technology would inevitably have been compromised. Instead of patent protection, the United States legislature could guarantee software an adequate legal protection through copyrights.

Cite as Giovanna Massarotto, *Open Source Paradigm: Beyond the Solution to the Software Patentability Debate*, 15 J. MARSHALL REV. INTELL. PROP. L. 647 (2016).

OPEN SOURCE PARADIGM: BEYOND THE SOLUTION TO THE SOFTWARE PATENTABILITY DEBATE

GIOVANNA MASSAROTTO

OPEN SOURCE PARADIGM: BEYOND THE SOLUTION TO THE SOFTWARE PATENTABILITY DEBATE

GIOVANNA MASSAROTTO*

I. INTRODUCTION

In 2016, innovation is the key motor of economic growth. Maintaining competition and economic incentives for companies to invest in new technologies is fundamental to preserve innovation. Historically, patent law and antitrust law conflict in protecting competition and maintaining economic incentives to innovate.[1] While patent law grants to the patent holder the right to exclude, essentially a legal monopoly of the invention,[2] antitrust law prohibits monopolistic behavior.[3]

Thus, the big challenge for both antitrust and patent law is to find an efficient compromise. To resolve this conflict, regulators must fully understand the specific circumstances and needs of each industry's market. A recent issue exemplifying the tension between antitrust and patent law is the patentability of software. In the United States, software is patented, while generally in Europe[4] and other countries, such as New Zealand,[5] it is not. On December 6, 2013, the U.S. Supreme Court granted *certiorari* in a case that questioned the patentability of a software method and system.

My analysis starts with the current issue debated by the U.S. Supreme Court on software patentability. Then, I examine the subjects of open source software and open source licenses, which conflict with the theory of patent law. The open source movement supports the use of open source licenses for some or all software,[6] while

---

* © Giovanna Massarotto 2016. Academic Visitor at the University of Oxford, Oxford; Of Counsel Massarotto & Associati, Treviso, giovanna@massarotto.com. I would like to thank Professor Mark Patterson and Andrea Lively for their extremely helpful comments. This paper has been selected for the Eleventh Annual Conference organized by SIDE (Italian Society of Law & Economics)—Naples, Italy (18 December 2015).

[1] *See, e.g.*, Michael A. Carrier, *Resolving the Patent-Antitrust Paradox Through Tripartite Innovation*, 56 VAND. L. REV. 1047 (2003).

[2] 35 U.S.C. § 154. *See, e.g.*, Greg Lastowka, *Inovative Copyright*, 109 MICH. L. REV. 1011, 1012 (2011).

[3] Sherman Antitrust Act, 15 U.S.C. § 2 (1890).

[4] In 2005, the European Parliament rejected legislation to allow software patents. Article 52 of the European Patent Convention specifies that "programs for computers" are not patentable. Administrative Council of the European Patent Organization, The European Patent Convention, O.J. EPO 2001, (June 28, 2001), *available at* http://www.epo.org/law-practice/legal-texts/html/epc/2013/e/ar52.html.

[5] In 2013, New Zealand passed legislation to ban software patents. New Zealand, Patent Act, n. 68 (2013) http://www.legislation.govt.nz/act/public/2013/0068/latest/DLM1419043.html.

[6] *See, e.g.*, Katherine Noyes, *GNU GPL Creator Richard Stallman on the Meaning of 'Free'*, LINUXINSIDER, (Oct. 15, 2007), http://www.linuxinsider.com/story/59780.html; David McGowan, *Legal Implications of Opensource Software*, 2001 U. ILL. L. REV. 241 (2001). "Unlike the traditional producers of computer software—Microsoft, for example—open-source software is often developed by computer programmers from all over the world, each submitting contributions to the code, and distributed without charge or for a minimal fee." *Id.*

other technology corporations, such as Microsoft and Apple, are more in favor of patenting software and protecting patent law.

Specifically, in Part II I discuss the most relevant U.S. case law and the recent Supreme Court case, *Alice Corp. v. CLS Bank Int'l.,*[7] on the patentability of software. In Part III, I analyze the thoughts of the current most influential computer scientists on the software patentability debate. To support the fact that software is not eligible for patent protection, I also discuss how Professor Noam Chomsky explained languages and how it relates to the underlying programming language of software.

In Part IV, I examine the implication of open source licensing and clarify the economic incentives to invest in open source software. Finally, in Part V, I identify copyright as the best legal protection for software.

Reflecting on similar issues is fundamental to helping lawmakers and courts promote competition and technological innovation. The common goal is to identify and implement the most appropriate and efficient economic/social paradigm. Perhaps such a standard could be identified in a regulated *open source paradigm*.

II. US PATENT LAW AND THE SOFTWARE PATENTABILITY DEBATE

In 2005, the European Parliament rejected the law that would have allowed software patents.[8] In contrast, over the past twenty years the intellectual property rights regime in the U.S. has significantly changed. Between 1972 and 1981, three Supreme Court decisions identified strict limits on patenting software. However, in 1982 around 1,200 software patents were granted; today the annual average is around 40,000 software patents.[9]

Generally, the U.S. did not have strict criteria for patents, which allowed people to patent, among other items, software and business models.[10] The U.S.'s decision to relax the patent criteria seems to stem from increasing foreign competition.[11]

Further, in 1982 Congress stated that the Federal Circuit Court of Appeals had jurisdiction over all appeals in patent lawsuits to unify the application of patent law.[12] The Federal Circuit soon became the specialized court for patent law taking

---

[7] Alice Corp. v. CLS Bank Int'l., 134 S. Ct. 2347 (2014).

[8] European Parliament sends software patent packing, (Jul. 19, 2005) http://ec.europa.eu/research/infocentre/article_en.cfm?id=/research/headlines/news/article_05_07_20_en.html&item=&artid=.

[9] Richard Wolf, *Supreme Court sets strict standard for computer patents*, USA TODAY, (Jun. 19, 2014), http://www.usatoday.com/story/money/business/2014/06/19/supreme-court-computer-patent/9190199/.

[10] *See, e.g.*, Carl Shapiro, *Patent System Reform: Economic Analysis and Critique*, 19 BERKELEY TECH. L. J. 1017 ("[c]omplaints regarding the patent system typically allege that the U.S. Patent and Trademark Office (USPTO) issues many questionable patents, for example, patents that are likely to be invalid or contain over broad claims").

[11] Benjamin Coriat & Fabienne Orsi, *Establishing a New Intellectual Property Rights Regime in the United States: Origins, Content and Problems*, 31 RES. POL'Y 1491 (2002).

[12] *See, e.g.*, Meredith Martin Addy, *Is the Federal Circuit Ready to Accept Plenary Authority for Patent Appeals?*, 4 J. MARSHALL REV. INTELL. PROP. L. 583 (2005); Rochelle Cooper Dreyfuss, *The Federal Circuit: a Case Study in Specialized Courts*, 64 N.Y.U. L. REV. 1 (1989); Timothy B. Lee, *Everything you need to know about patents*, VOX, (Oct. 7, 2014) *available at* http://www.vox.com/cards/patent-reform.

the role previously covered by the U.S. Court of Customs and Patent Appeals ("CCPA"). The Federal Circuit had to follow the precedents of the CCPA in applying patent law.[13]

The new role covered by the Federal Circuit Court of Appeals was not without criticism. In 2003, the Federal Trade Commission issued the report *To Promote Innovation: the Proper Balance of Competition and Patent Law and Policy*,[14] criticizing the "pro-patent stance" of the Federal Circuit.[15] The Federal Trade Commission noted that in the software industry "firms can require access to dozens, hundreds, or even thousands of patents to produce just one commercial product."[16] In 1981, the Supreme Court remarked in *Diamond v. Diehr, et al.*[17] that the only categories not patentable are "laws of nature, natural phenomena, and abstract ideas."[18] In 2012, the Federal Circuit Court of Appeals in *CLS Bank Int'l v. Alice Corp*. made the same statement supporting the patentability of Alice's invention.[19] Then, the Federal Circuit granted a petition for rehearing *en banc* and affirmed the District Court's judgment recognizing that Alice's invention was patent-ineligible.[20] The invention concerned a computerized trading platform that serves to conduct financial transactions where a third party negotiates obligations between a first and a second party to eliminate the settlement risk.[21]

Thus, important questions to ask are: When is software eligible for patenting? What are the standards for patenting software?

A. The Supreme Court Decisions on Software Patentability

Although a majority of the judges on the Federal Circuit agree that method claims do not recite patent-eligible subject matter, no majority of those judges agree as to the legal rationale for that conclusion.[22]

This is what Chief Judge Rader observed in *CLS Bank*, commenting on the lack of clarity in software patentability. As the following Supreme Court decisions reveal, the issue of whether software is patent-eligible is not definitively determined. The

---

[13] Addy, *supra* note 12, at 583.

[14] FEDERAL TRADE COMMISSION, TO PROMOTE INNOVATION: THE PROPER BALANCE OF COMPETITION AND PATENT LAW AND POLICY, A REPORT BY THE FEDERAL TRADE COMMISSION (2003), *available at* http://www.ftc.gov/sites/default/files/documents/reports/promote-innovation-proper-balance-competition-and-patent-law-and-policy/innovationrpt.pdf.

[15] *Id. See also* Thomas P. Burke, *Software Patent Protection: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1132 (1994) ("[f]rom 1982 to 1992, there were one hundred and fifty-two reported patent infringement cases with damage awards totaling $1.73 billion").

[16] FEDERAL TRADE COMMISSION, *supra* note 14, at 32.

[17] 450 U.S. 175 (1981).

[18] *Id.* at 185.

[19] CLS Bank Int'l v. Alice Corp., 685 F.3d 1341, 1352 (Fed. Cir. 2012), *vacated*, 484 F. App'x 559 (Fed. Cir. 2012).

[20] CLS Bank Int'l v. Alice Corp., 717 F. 3d 1269, 1274 (Fed. Cir. 2013). *See also*, Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347, 2353 (2014).

[21] *CLS Bank*, 685 F.3d at 1341.

[22] *CLS Bank*, 717 F.3d at 1292 n.1 (Fed. Cir. 2013) (Rader, C.J., concurring in part and dissenting in part), *cert. granted*, 134 S. Ct. 734 (2013).

latest Supreme Court decision on this topic, again, did not resolve the issue by taking a clear position.

1. Gottschalk v. Benson (1972)

In 1972, the Supreme Court decided on the patentability of a method for converting binary-coded decimal ("BCD") numerals into pure binary numerals—namely an algorithm. The claims purported "to cover any use of the claimed method"[23] without being limited to any art or technology, apparatus or machinery, or particular end use.[24]

Although both the U.S. Patent and Trademark Office and the Board of Patent Appeals and Interferences rejected the patent application, the CCPA reversed the rejection. In reply to the CCPA's decision, the Commissioner of Patent and Trademark decided to file a petition for writ of certiorari to determine whether the claim presented patentable subject matter under § 101.[25]

The Supreme Court recognized that "[a] procedure for solving a given type of mathematical problem is known as an 'algorithm,'" and that "[t]he procedures set forth in the present claims are of that kind." In other words, these procedures represent a useful generalized formulation used for resolving mathematical problems of converting one form of numerical representation to another by programs.[26]

Thus, the Supreme Court concluded that the case at hand involved a mathematical formula and if the judgment of the CCPA, that embraced the claim, were affirmed, "the patent would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself."[27]

Therefore, the Supreme Court reversed the CCPA's judgment and rejected the proposal that such programs were patentable.

2. Parker v. Flook (1978)

The patent in this suit claimed a method "for Updating Alarm Limits," where the only new element was the applied mathematical formula.[28] According to the Court, "[t]he only difference between the conventional methods of changing alarm limits and that described in respondent's application rest[ed] in the second step—the mathematical algorithm or formula." Similar to *Gottschalk*, the patent examiner and the Board of Patent Appeals and Interferences rejected the patent application,[29] while the CCPA reversed their previous decisions.[30]

---

[23] Gottschalk v. Benson, 409 U.S. 63, 64 (1972).

[24] *Id.*

[25] 35 U.S.C. § 101. "[W]hoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title." *Id.*

[26] *Id.*

[27] *Id.* at 72.

[28] Parker v. Flook, 437 U.S. 584, 585 (1978).

[29] *Id.* at 587.

[30] *Id.*

The Supreme Court, in deciding on the patentability of this method, used the same analysis adopted in the *Mackay Radio & Telegraph Co, Inc. v. Radio Corp. of Am.*[31] and *Funk Bros. Seed Co. v. Kalo Inoculant Co.*[32]. The Court recognized that "[w]hether the algorithm was . . . known or unknown at the time of the claimed invention, as one of the 'basic tools of scientific and technological work,' . . . it is treated as though it were a familiar part of the prior art."[33] This is the same reason why the Court rejected "Samuel Morse's broad claim covering any use of electromagnetism for printing intelligible signs, characters, or letters at a distance."[34]

In sum, the Supreme Court again reversed the CCPA's judgment because the claim at hand simply provided a formula for computing an updated alarm limit, it was ineligible for patenting.[35]

3. Diamond v. Diehr (1981)

In 1981, the Supreme Court granted certiorari to decide whether "a process for curing synthetic rubber which includes in several of its steps the use of a mathematical formula and a programmed digital computer is patentable subject matter under 35 U.S.C. § 101."[36] The Supreme Court observed that, unlike *Benson* and *Parker*, the respondents do not seek to patent a mathematical formula but seek patent protection for a process of curing synthetic rubber.[37] Specifically, the claimed invention concerned "a process for molding raw, uncured synthetic rubber into cured precision products."[38]

Because "a physical and chemical process for molding precision synthetic rubber products falls within the § 101 categories of possibly patentable subject matter,"[39] the Supreme Court recognized the patentability of the claims. In particular, the Supreme Court recognized that "[i]n determining the eligibility of respondents' claimed process for patent protection under § 101, their claims must be considered as a whole."[40] In a process claim this is particularly true. The Supreme Court specified that, although all the constituents of the combination in a process were well known and in common use before the combination was made, a new combination of process steps "may be patentable."[41] By doing so, the Supreme Court alleged an important principle: a process that includes mathematical formulae and a programmed digital computer may be patent-eligible.

---

[31] Mackay Radio & Telegraph Co., Inc. v. Radio Corp. of Am., 59 S. Ct. 427 (1939).
[32] Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127 (1948).
[33] *Parker*, 437 U.S. at 592.
[34] *Id.*
[35] *Id.* at 594-96.
[36] Diamond v. Diehr, 450 U.S. 175, 177 (1981).
[37] *Id.* at 187.
[38] *Id.* at 177.
[39] *Id.* at 184.
[40] *Id.* at 189.
[41] *Id.*

4. Bilski v. Kappos (2010)

Here, the claimed invention concerned "a procedure for instructing buyers and sellers how to protect against the risk of price fluctuations in a discrete section of the economy."[42] The issue at hand turned on "whether a patent can be issued for a claimed invention designed for the business world."[43]

The Supreme Court recognized that the Federal Circuit Court of Appeals incorrectly alleged that this Court has endorsed the machine-or-transformation test as exclusive.[44] According to the Supreme Court, the machine-or-transformation test represents only an important clue in deciding whether some claimed inventions are processes under § 101.[45] A business method like the one here, however, might be patentable. Specifically, the Supreme Court recognized that "the Patent Act leaves open the possibility that there are at least some processes that can be fairly described as business methods that are patentable subject matter under § 101."[46] In this case, however, the claimed business methods were not patentable and "the patent application has been rejected under [the] Supreme Court's precedent on the un-patentability of abstract ideas."[47]

5. Mayo v. Prometheus (2012)

Although the Supreme Court has attempted to provide guidance on software patentability, in cases that concern this issue, courts engage in a difficult process of deciding whether a claim presents patentable subject matter under § 101.[48] In *Mayo v. Prometheus*,[49] the Supreme Court decided whether a diagnostic test was patent eligible.

Prometheus Laboratories Inc. ("Prometheus") sold diagnostic tests, which included some patented processes. Mayo Clinic Rochester and Mayo Collaborative Services (collectively "Mayo") bought and employed those tests, but Mayo decided to use its own test in 2004. Prometheus sued Mayo for patent infringement and Mayo counterclaimed that Prometheus' patents were invalid because they effectively claim natural law or phenomena.[50] Specifically, the claims involved "the precise correlations between metabolite levels and likely harm or ineffectiveness." [51]

The district court recognized that because Prometheus' patents claimed natural phenomena and natural law they were not patent-eligible.[52] Prometheus appealed the decision. Despite the Federal Circuit's reversal of the district court's decision, the

---

[42] Bilski v. Kappos, 130 S. Ct. 3218, 3223 (2010).
[43] *Id.*
[44] *Id.* at 3227.
[45] *Id.* at 3226.
[46] *Id.* at 3229.
[47] *Id.* at 3231.
[48] Dennis Crouch & Robert P. Merges, *Operating Efficiently Post-Bilski by Ordering Patent Doctrine Decision-Making*, 25 BERKELEY TECH. L. J. 1673 (2010).
[49] Mayo Collaborative v. Prometheus Labs., 132 S. Ct. 1289 (2012).
[50] *Id.* at 1296.
[51] *Id.* at 1295.
[52] *Id.* at 1296.

Supreme Court concluded that "the steps in the claimed processes (apart from the natural laws themselves) involve well-understood, routine, conversational activity previously engaged in by researchers in the field" and that "the claims [were] consequently invalid."[53]

6. Summary of the U.S. Case Law on Software Patentability

In sum, over the last forty years the Supreme Court has recognized that: (a) the only categories not patentable are "laws of nature, natural phenomena, and abstract ideas;"[54] (b) because an algorithm is based on a mathematical formula, algorithms are not patent-eligible; and (c) a new combination of process steps may be patentable although all the elements of a combination in a process were well known before the combination was made.[55]

However, despite the number of Supreme Court rulings on software patentability, the underlying issue is not yet resolved. In 2013, the software patentability debate reappeared before the Supreme Court in *Alice Corp. v. Cls Bank Int'l.* Further, each year, the U.S. Patent and Trademark Office grants an average of approximately 40,000 software patents.[56] At this point, we must ask from whence the confusion comes: is the Supreme Court unclear in defining software patentability criteria; or are the activities of the U.S. Patent and Trademark Office inconsistent with Supreme Court case law? Whatever the response, the law must be clear and public enforcers must be held to application of legal provisions and principles; not broad use of discretion.

B. Supreme Court—Certiorari—Alice Corp. v. CLS Bank Int'l

On December 6, 2013, the U.S. Supreme Court granted certiorari in *Alice Corp. v. CLS Bank Int'l.*[57] The question posed by Alice, the patent owner, was:

> Whether claims to computer-implemented inventions—including claims to systems and machines, processes, and items of manufacture—are directed to patent-eligible subject matter within the meaning of 35 U.S.C. § 101 as interpreted by this Court?[58]

---

[53] *Id.* at 1305.

[54] *Diamond*, 450 U.S. at 185.

[55] *Id.*

[56] *See, e.g.*, Brian J. Love, *Requests For Comments on Enhancing Patent Quality*, 2 n.5 (May 6, 2015) http://digitalcommons.law.scu.edu/cgi/viewcontent.cgi?article=1883&context=facpubs (*citing* Christina Mulligan & Timothy B. Lee, *Scaling the Patent System*, N.Y.U. ANN. SURV. AM. L. 289 (2012).

[57] *CLS Bank*, 717 F.3d at 1292 n.1 (Rader, C.J., concurring in part and dissenting in part), *cert. granted*, 134 S. Ct. 734 (2013).

[58] Brief for Petitioner at (i), Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347 (2013) (No. 13-298), 2014 WL 262088.

In other words, the Supreme Court was questioned on the patent-eligibility of some kinds of software. In this case, Alice Corp. Pty. ("Alice") owned patents related to "the management of risk relating to specified, yet unknown, future events."[59] Specifically, the patents in this suit concerned a computerized trading platform that serves to conduct financial transactions where a third party negotiates obligations between a first and a second party to eliminate the settlement risk ('479, '510, and '720 patents).

On May 24, 2007, CLS Bank ("CLS"), a bank that developed software for its own use, filed a suit against Alice seeking a declaratory judgment of non-infringement, invalidity, and unenforceability as to the '479, '510, and '720 patents. According to CLS, Alice's patents were ineligible because they merely recited abstract ideas concerning some fundamental economic concepts. In response, Alice counterclaimed alleging that CLS infringed its patents.

While the district court held that the claims at issue were invalid under § 101, on July 9, 2012 the Court of Appeals for the Federal Circuit recognized that the asserted claims of Alice's patents were all patent-eligible under § 101.[60] According to the district court, the claims at issue "would preempt the use of the abstract concept of employing a neutral intermediary to facilitate simultaneous exchange of obligations in order to minimize risk on any computer, which is, as a practical matter, how these processes are likely to be applied."[61]

Conversely, the Federal Circuit, reversing the district court's judgment, argued that the claims at issue were not "mere 'abstract ideas' but rather [were] directed to practical applications of invention falling within the categories of patent-eligible subject matter defined by 35 U.S.C. § 101."[62]

Nevertheless, in October 2012, the Court of Appeals for the Federal Circuit granted a petition for rehearing *en banc* filed by CLS.[63] This time, the Federal Circuit agreed with the district court's judgment recognizing that "the asserted method, computer-readable medium, and system claims of Alice's '479, 510', '720, and '375 patents [were] invalid under § 101 for failure to recite patent-eligible subject matter."[64]

The Federal Circuit summarized the steps of patent-eligibility analysis as follows. First, we should ask "whether the claimed invention is a process, machine, manufacture, or composition of matter"[65] and "[i]f not, the claim is ineligible under § 101."[66] Second, "[i]f the invention falls within one of the statutory categories, we must then determine whether any of the three judicial exceptions nonetheless bars such a claim—is the claim drawn to a patenting-eligible law of nature, natural

---

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

[60] CLS Bank Int'l v. Alice Corp., 685 F.3d 1341, 1352 (Fed. Cir. 2012), *vacated*, 484 F. App'x 559 (Fed. Cir. 2012).

[61] CLS Bank Int'l v. Alice Corp., 768 F.Supp.2d 221, 252 (D.C. Cir. 2011).

[62] *CLS Bank*, 685 F.3d at 1343.

[63] *CLS Bank*, 484 Fed. Appx. 559 (Fed. Cir. 2012); *see* FED. R. APP. P. 35(a) ("*When Hearing or Rehearing En Banc May Be Ordered*. A majority of the circuit judges who are in regular active service and who are not disqualified may order that an appeal or other proceeding be heard or reheard by the court of appeals *en banc*.")

[64] *CLS Bank*, 717 F. 3d at 1292 (Fed. Cir. 2013).

[65] *Id.* at 1277.

[66] *Id.*

phenomenon, or abstract idea?" And, "[i]f so, the claim is not patent-eligible. Only claims that pass both inquiries satisfy § 101."[67]

The Supreme Court affirmed the Court of Appeal's ruling concluding that "[b]ecause [Alice]'s system and media claims add nothing of substance to the underlying abstract idea, [the Supreme Court held] that they too are patent-ineligible under § 101."[68]

Although Alice's question concerned whether all claims to computer-implemented inventions were patent-eligible, the Supreme Court's decision did not answer it. The Supreme Court only decided whether Alice's computer-implemented invention was patent-eligible, a narrow question, applying the two-step framework set by the Court in *Mayo*.[69] The Supreme Court only scratched the surface of the problem. The issue of software patentability was not entirely resolved and will likely continue to be litigated.

C. Patent Trolls and Patent Privateering Phenomena

With respect to patent litigation, it is important to bear in mind that the practice of so-called "patent trolls" raises concerns in the United States. But what does "patent troll" mean? As Professor Mark A. Lemley clarified, patent trolls are patent holders that do not practice patents; but rather, their main business is gathering money from other companies that infringe their patents.[70] Since patent trolls do not create any products or services, they are also known as Patent Assertion Entities ("PAEs") or Non-Practicing Entities ("NPEs").[71]

In several cases, patent trolls buy patents from the original patent holder, and later engage in lawsuits against competing firms of the original patent owner. This strategy, where the patent troll acts as a privateer is also called *patent privateering*.[72]

Patent trolls and the patent privateering phenomena is particularly challenging in the context of software.

According to the patent scholars James Bessen and Micheal J. Meur, patent troll litigation cost the U.S. economy $29 billion in 2011, fueling a widespread feeling that there is too much intellectual property litigation.[73] Patent litigation has required defendants to pay tens of billions of dollars per year, and many of these lawsuits have

---

[67] *Id.*

[68] *Alice*, 134 S. Ct. at 2360.

[69] *Id.* at 2353-2356.

[70] Mark A. Lemley & A. Douglas Melamed, *Missing the Foster for the Trolls*, 113 COLUMBIA L. REV. 1001 (2013).

[71] Thibault Schrepel, *Patent Privateering, or Patents as Weapons*, LE CONCORRENTIALISTE 1, 2 (2014).

[72] *Id.*; *see also* John M. Golden, *Patent Privateers: Private Enforcement's Historical Survivors*, 26 HARVARD J. L. & TECH. 546, 589 (2013).

[73] James Bessen & Micheal J. Meur, *The Direct Costs from NPE Disputes*, 99 CORNELL L. REV. 387 (2014); *Time to Fix Patents, Ideas fuel the economy: Today's patent systems are a rotten way of rewarding them*, THE ECONOMIST (Aug. 8, 2015); *A question of utility, Patents are protected by governments because they are held to promote innovation. But there is plenty of evidence that they do not*, THE ECONOMIST, (Aug. 8, 2015).

concerned software patents.[74] Open source developers and users are particularly vulnerable to patent troll attacks, whereas large firms protect themselves by purchasing a huge number of patents.[75]

Therefore, legislators and courts should identify clear standards or suggest an appropriate paradigm in the software industry to foster and guarantee technology development that remains consistent with intellectual property and competition law purposes.

Finding a clear solution on software patentability and deciding whether all claims to computer-implemented inventions are patent-eligible is crucial not only for computer programmers but also for the economy as a whole. Software likely represents the most important element in computer science and a fundamental building block of technology development.

III. COMPUTER SCIENTISTS AND THE STUDY OF COMPUTER LANGUAGE

Having analyzed U.S. case law on software patentability, patent trolls and patent privateering phenomena, I next explain what the most influential computer scientists say about software patents and software patentability. Is software patentability good for fostering innovation? Software and computer technology are based on a number of computer languages. Patenting software would mean patenting languages and algorithms underlying the software. What does computer language mean? HTML, URLs, and HTTP, which make the Web work, include a set of different computer languages.[76] HTML (HyperText Markup Language), for example, represents the standard markup language used to create webpages. Therefore, what if the Web and its languages had been patented? This and other similar questions are explored in this section.

---

[74] Timothy B. Lee, *Everything you need to know about software patents*, VOX, (Oct. 7, 2014), http://www.vox.com/cards/software-patents. *Rockstar v. Google* is only an example of patent litigation that concerns software patentability. *See* Rayan Davis, *Mass. AG Latest To Set Sights On Patent Trolls*, LAW360, (Nov. 6, 2013), http://www.law360.com/ip/articles/486777/mass-ag-latest-to-set-sights-on-patent-trolls. *See, e.g.*, Steven J. Vaughan-Nichols, *Biggest patent win ever? Microsoft's billion dollar a year Samsung deal*, ZD NET, (Oct. 5, 2014), http://www.zdnet.com/article/biggest-patent-win-ever-microsofts-billion-dollar-a-year-samsung-deal/ ("In 2013 alone, Microsoft made a billion dollars from its Samsung Android patent licensing deal alone. In that same year, Microsoft profited from its Android patents to a tune of about $3.4 billion.").

[75] James Boyle, *Open Source Innovation, Patent, Injunction, and the Public Interest*, 11 DUKE L. & TECH. REV. 30, 33 (2013); *see also* Bruce Perens, *Preparing for the Intellectual Property Offensive* (1998), http://www.linuxworld.comllinuxworld. *See also* Stuart J.H. Graham & David C. Mowery, *The Use of USPTO "Continuation" Applications in the Patenting of Software: Implications for Free and Open Source*, 27 LAW & POL'Y 128, 140 (2003).

[76] Tim Berners-Lee, *Speech and the Future*, SPEECH TECK NEW YORK, (Sept. 24, 2004), http://www.w3.org/2004/Talks/0914-tbl-speech/text.

A. What Computer Scientists think about Software Patentability?

1. The "Father" of Analysis of Algorithms—Prof. Donald Knuth

In 1994, Prof. Donald Knuth, well-known in the computer science world as the "father" of analysis of algorithms,[77] addressed a letter to the Commissioner of Patents and Trademarks asking to reconsider the current policy of granting patents for computational processes.[78] Although the Supreme Court's rulings in the 1970s recognized that software was not eligible for patent protection;[79] since 1980 patent courts and the Patent and Trademark Office have granted hundreds of thousands of software patents.[80]

According to Prof. Knuth, this change of approach has harmed society.[81] Prof. Knuth compared algorithms with words, alleging that because algorithms are the fundamental building blocks needed to make software, "algorithms are exactly as basic to software as words are to writers."[82] Thus, patenting software and the related algorithm would limit software development.

2. The Founder of the Free Software Foundation—Richard M. Stallman

Similarly, Richard M. Stallman, a famous computer programmer who founded the Free Software Foundation ("FSF"), compared algorithms or techniques to a series of musical notes or a chord progression.[83] Patenting a series of musical notes or a chord progression would mean forcing composers to purchase a "musical sequence license."[84] An algorithm, Stallman clarifies, is a method for solving a problem. Algorithms are necessary for creating software. When software and its algorithms are patented, it implies that in order to use that algorithm you would need the owner's authorization. Thus, patenting software and related algorithms would limit

---

[77] *See, e.g.* Donald Knuth, *Algorithmic Thinking and Mathematical Thinking*, 92 THE AMERICAN MATHEMATICAL MONTHLY 170 (1985). According to Prof. Knuth, "computer science is primarily the study of algorithm." *Id.* He provided a much broader definition of algorithms than other computer scientists, considering an algorithm "as encompassing the whole range of concepts dealing with well-defined processes, including the structure of data that is being acted upon as well as the structure of the sequence of operations being performed." *Id.*

[78] Knuth, *Letter to the Patent Office From Professor Donald Knuth* (1994), http://progfree.org/Patents/knuth-to-pto.txt.

[79] The Supreme Court, for example, in *Benson* rejected an algorithm for converting binary-coded decimal numerals into pure binary form as an ineligible patent claims. According to the Court, the claimed patent was "in practical effect . . . a patent on the algorithm itself." 409 U.S. at 71-72, 93 S. Ct. 253. Similarly, in *Parker*, the Court recognized that a mathematical formula for computing "alarm limits" in a catalytic conversion process was a patent-ineligible abstract idea. 437 U.S. at 594-595.

[80] *See, e.g.*, Wlliam D. Wiese, *Death of a Myth: the Patenting of Internet Business Models After State Street Bank*, 4 MARQ. INTELL. PROP. L. REV. 17, 18 (2000); Burke, *supra* note 15, at 1157.

[81] Knuth, *supra* note 77.

[82] *Id.*

[83] Simson L. Garfinkel, Richard M. Stallman & Mitchell Kapor, *Why Patents are Bad for Software*, SCI. & TECH. (1991).

[84] *Id.*

the improvement of methods for solving problems and creating new software. Stallman defined the proprietary software social system as "unethical."[85]

3. The Founder of Linux Operating System—Linus Torvalds

Another well-known computer scientist, Linus Torvalds, the founder of the Linux operating system,[86] argued that the U.S. patent system has become so broken that it is hindering innovation.[87] Linus Torvalds required the United States to abolish software patents. Linux is an open source operating system that is used everywhere. For example, Linux powers Google, Facebook, Twitter, Amazon, air traffic control systems, the International Space Station, and nine out of the world's ten supercomputers run on Linux.[88]

What would happen if Linux had been patented? Would we be able, for example, to run Google or Facebook?

4. The Father of the Web—Tim Berners-Lee

Finally, in 1990 Tim Berners-Lee, the inventor of the World Wide Web and commonly referred to as father of the Internet, wrote the first simple specs of URLs (then UDIs), HTML and HTTP, which are based on a set of computer languages. By 1993, the Web was expanding very rapidly, and the reason the Web was spreading so fast was that there was no central control and no royalty fee. Anyone could start playing with it, browsing, running a server, or writing software without commitment and without ending up in the control of, or owing money to, any central company.[89]

The success of the Web implies that patentability of software and computer languages may not be the most economically efficient.

B. What If the Web Had Been Patented? The Microsoft Network

In January 1995, Bill Gates, the chairman of Microsoft, announced that his next business target was the Internet.[90] In August 24, 1995, Microsoft launched the Microsoft Network ("MSN") alongside Windows 1995.

---

[85] Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source*, 2 J. INDUSTRIAL ECON., 197, 198 (2002).

[86] International Data Corporation has estimated that in the personal computer operating system market, the open source program Linux, with a 200 percent annual growth rate, has between seven to twenty-one million users worldwide. *Id.* at 197.

[87] Dylan Love, *A Conversation with Linus Torvalds, Who Built the World's Most Robust Operating System and Gave it Away for Free*, BUSINESS INSIDER, (Jun. 7, 2014) http://www.businessinsider.com/linus-torvalds-qa-2014-6?IR=T.

[88] Linux Foundation, *Find your Path to Open Collaboration, The Future is Open*, 2 (2013), www.linuxfoundation.org/sites/main/files/linux_foundation_brochure.pdf.

[89] Berners-Lee, *supra* note 76.

[90] Philip Elmer-DeWitt, *Will Gates Get the Net?*, (Jan. 23, 1995), http://scripting.com/davenet/1995/01/23/billgatesvstheinternetpart.html.

Microsoft bought a minority stake in an Internet access company and built a nationwide network where customers would be able to dial into the Internet. Originally, MSN was integrated into the Windows Explorer file management program through an artificial folder-like graphical user interface in the new Windows 95.[91] The premise was that Gates, whose software ran on nine out of ten personal computers, would have done to the Internet what he did to the Personal Computer ("PC") market. Microsoft would have extended its domination, leveraging its control in the software and PC market, and imposed a new vertical strategy in the computer industry. But Gates did not seem to be able to control the Internet with the launch of MSN.

In contrast to the free web developed by Tim Berners-Lee, Microsoft's consumers paid for MSN by the month or hour.[92]

According to Brad Templeton, president of ClariNet Communication Corp., the first Internet (and for a long time the largest) electronic newspaper, Microsoft could not control the central network operating system.[93] The Internet was dedicated to open, nonproprietary software systems. For example, "a week after the Internet community discovered that the GIF ("Graphics Interchange Format") system used to exchange pictures over the network contained a patented compression scheme and that the patent holder was demanding royalty payments,"[94] somebody introduced a free alternative product. Hence, a graphics-exchange format that worked like GIF, but patent-free, was quickly developed.[95]

Microsoft was not the only company that began its own Internet network.[96] In 2011, Francis Gurry, the Director General of the UN's World Intellectual Property Organization ("WIPO"), alleged that "the Web would have been better off if it had been locked away in patents, and if every user of the Web . . . needed to pay a license fee to use it."[97] However, MSN and other proprietary Internet networks did not have the hoped-for success and did not compromise the development of the free Web. The latter soon became the fastest growing network, vital to our economy[98], and Gates's strategy to dominate the Internet failed.[99] As demonstrated by this scenario, Gurry's suggestion is not a feasible one.

Focusing back on our initial question: What would have happened if the Web had been patented? Of course, the Web's innovation level would have been significantly

---

[91] James Gleick, *Making Microsoft Safe for Capitalism*, N.Y. TIMES, (November 5, 1995) http://www.around.com/microsoft.html.

[92] *Id.*

[93] Elmer-DeWitt, *supra* note 90.

[94] *Id.*

[95] *Id.*

[96] *Id.*

[97] Cory Doctorow, *WIPO boss: the Web would have been better if it was patented and its users had to pay license fees,* (Oct. 8, 2011), http://boingboing.net/2011/10/08/wipo-boss-the-web-would-have-been-better-if-it-was-patented-and-its-users-had-to-pay-license-fees.html.

[98] Mark A. Lemley & Lawrence Lessing, *The End of End-to-End: Preserving the Architecture of the Internet in the Broadband Era*, Research Paper No. 2000-19, 61 *available at* http://cyberlaw.stanford.edu/e2e/papers/Lemley_Lessig_e2epaper.pdf.

[99] Donelle Gan, *May 26, 1995: Gates, Microsoft Jump on 'Internet Tidal Wave'*, WIRED, (May 26, 2010), http://www.wired.com/2010/05/0526bill-gates-internet-memo/.

limited, and we would likely not have Google Chrome,[100] Facebook, or Android. People would have had to pay a royalty fee to use the Web and consumer welfare would have been compromised.

In 1995, Tim Berners-Lee said in a talk at MIT:

> I had (and still have) a dream that the web could be less of a television channel and more of an interactive sea of shared knowledge. I imagine it immersing us as a warm, friendly environment made of the things we and our friends have seen, heard, believe or have figured out. I would like it to bring our friends and colleagues closer, in that by working on this knowledge together we can come to better understandings.[101]

In 2015, I am happy to recognize that Tim Berners-Lee's dream came true. As the father of the Internet argued, the fact that there is no central control in the Web and no royalty fees represents its success.[102]

However, World Wide Web Consortium ("W3C") critics object that the W3C, led by Tim Berners-Lee and Jeffrey Jaffe,[103] controls the Web and that W3C is concentrated among large firms and software sellers.[104] In 1994, Berners-Lee founded W3C, an international community created to realize the full potential of the Web by identifying standards such as the HTML Table and Pics. The W3C is a standard organization rather than the Web's owner. W3C has learned from past experiences the importance of having the browser developers' support in identifying new web standards and developing the web as a whole.[105] Because W3C does not control the Web, W3C cannot force browser developers to conform to its standards. But, involving large firms in the standardization process is fundamental to creating shared and evenly-developed standards.

The Web is open and universal. As Berners-Lee observed, you cannot pretend that something is universal if someone keeps control of it.[106] Thus, the W3C's goal is far from imposing any form of control on the web.

---

[100] Mike Masnick, *What If Tim Berners-Lee Had Patented The Web?,* TECHDIRT, (Aug. 11, 2011), https://www.techdirt.com/articles/20110811/10245715476/what-if-tim-berners-lee-had-patented-web.shtml.

[101] Tim Berners-Lee, *Hypertext and Our Collective Destiny*, (Oct. 12, 1995), http://www.w3.org/Talks/9510_Bush/Talk.html.

[102] Berners-Lee, *supra* note 76.

[103] W3C, *About W3C*, http://www.w3.org/Consortium/, (last visited, Nov. 29, 2015).

[104] Paul Festa, *Critics clamor for Web services standards*, CNET, (July 10, 2002) http://www.cnet.com/news/critics-clamor-for-web-services-standards/.

[105] Ryan Paul, *Tim Berners-Lee talks about W3C reform and reinventing HTML*, (30, Oct. 2006), http://arstechnica.com/uncategorized/2006/10/8101/.

[106] Tim Berners-Lee, *Frequently asked questions*, http://www.w3.org/People/Berners-Lee/FAQ.html#Roles (last visited, Nov. 19, 2015).

C. What does Computer Language Mean? Noam Chomsky and the Hierarchy of Grammars

Despite Prof. Knuth's observation that "algorithms are exactly as basic to software as words are to writers;"[107] and consistent with Prof. Noam Chomsky's studies of languages, algorithms look more like sentences than only a word. Algorithms are the foundation for computer languages. Patenting software would imply patenting the underlying languages and algorithms. But what does computer language mean? What is the difference between human languages, such as English, and computer languages?

Prof. Chomsky's studies of languages have been fundamental for computer scientists who use them for describing the syntax of the programming language.[108] Prof. Chomsky defines language as "a collection of sentences of finite length all constructed from a finite alphabet of symbols,"[109] and grammar as "a device of some sort for producing the sentences of the language under analysis."

In 1957, Prof. Chomsky wrote a book on syntactic structures,[110] which provided[111] "a general method for selecting a grammar for each language,"[112] and explained the complexity of languages through the hierarchy of languages and grammar.[113]

All forms of information—like pictures, numbers, names and sounds—can be ascribable to strings,[114] and the set of strings make up a language.[115] This concept is central to the computer science community which uses programming language to communicate instructions to computers.

Conversely, grammar is a set of rules that provides "a recursive enumeration"[116] of the strings associated with the language. Prof. Chomsky, in his hierarchy of grammars, identified four different levels of grammars: 1) Type-0 grammars (unrestricted grammars) comprise all formal grammars; 2) Type-1 grammars (context-sensitive grammars) creates the context-sensitive language; 3) Type-2 grammars (context-free grammars) includes the context-free languages; and 4) Type-3 grammars (regular grammars) which generate the regular languages generally applied to identify search patterns and the lexical structure of

---

[107] Knuth, *supra* note 77.

[108] Tao Jiang, Ming Li, Bala Ravikumar, & Kenneth W. Regan, *Formal Grammars and Languages*, 1 *available at* http://www.cs.ucr.edu/~jiang/cs215/tao-new.pdf.

[109] Massachusetts Institute of Technology, Noam Chomsky, http://web.mit.edu/linguistics/people/faculty/chomsky/, (last visited, Apr. 17, 20014*). See also*, Noam Chomsky, Bios, http://www.chomsky.info/bios.htm, (last visited, Apr. 17, 2014).

[110] NOAM CHOMSKY, SYNTACTIC STRUCTURES at 11 (1957).

[111] Noam Chomsky is a professor and one of the most important scholars of linguistics and the study of language.

[112] CHOMSKY, supra note 110, at *vii*.

[113] *Id.*

[114] Riitta Alkula, *From Plain Character Strings to Meaningful Words: Producing Better Full Text Databases for Inflectional and Compounding Languages with Morphological Analysis Software*, 4 INFORMATION RETRIEVAL 195, 196 (2001). ("Each word form is an individual character string and, consequently, a separate index entry (for example, dog and dogs are separate entries).").

[115] Dana Angluin, *Finding Patterns Common to a Set of Strings*, 21 J. COMPUTER & SYSTEM SCIENCES 41 (1980).

[116] Noam Chomsky & Marcel-Paul Schützenberger, *The Algebraic Theory of Context-Free Languages*, 35 COMPUTER PROGRAMMING AND FORMAL SYS. 118 (1963).

programming language.[117] Every Type-3 language is also Type-2, 1, and 0, and every Type-2 is also 1 and 0.

The context-free languages, Type-2, describe the programming language of computers,[118] and Type-3 is, for example, English or French. Computer scientists used Context-free grammar to build compilers to verify the syntax of computer programs.[119] Context-free grammar concerns a finite set of grammar rules that are used to create strings.[120]

Computers are designed to perform only a specific category and number of operations and not to perform any operation, as human languages are able to perform.[121] A computer executes a specific instruction that determines the necessary data, device, or mechanism to perform the operation.[122] A program is a sequence of instructions used to solve a specific problem. In computers, the string of zeroes and ones represents an instruction. Since it is impractical for humans to perform strings of zeroes and ones, computer programs—software—translate it into "true machine words and load [it] into memory ready for execution."[123] In sum, software allows humans to understand the output of machine language instructions. Software is a set of instructions by which programmers tell computers what to do, implying that the scope of software patents is very broad. For example, Amazon's '1-Click Ordering' is considered a software patent, namely "[a] method and system for placing an order to purchase an item via the Internet."[124]

Hence, software uses a specific language to resolve a problem, and algorithms can be compared with the sentences of such language.[125] When we speak we can use different words and orders of words to repeat what someone else has already said or to even explain new and different concepts. Computer programs sometimes include millions of lines of code, and a computer programmer using these lines could infringe a few patented lines of code. Patenting software would risk implicitly limiting programmers when using their own language (a set of strings) to communicate with computers, and even the Federal Trade Commission has recognized such risk.[126]

Programming language is classified as Type-2 language and grammar. Chomsky used the term Type-2 grammar to refer to context-free grammar; this is exactly how programming grammar and languages should be—free.

---

[117] Noam Chomsky, *On Certain Formal Properties of Grammars*, INFORMATION AND CONTROL 2, 167-167 (1959).

[118] Chomsky Hierarchy, http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Chomsky_hierarchy.html.

[119] Jiang, Li, Ravikumar, & Regan, *supra* note 108, at 1.

[120] Noam Chomsky & Schützenberg, *supra* note 116.

[121] M. Goldstein, *Computer Languages*, 72 AM. MATHEMATICAL MONTHLY 141 (1965).

[122] *Id.*

[123] *Id.* at 143 (1965).

[124] Method and Sys. for Placing a Purchase Ord. Via a Comm. Network, U.S. Patent No. 5,960,411 (filed Sept. 12, 1997) (issued Sept. 28, 1999); *see also* Lee, *supra* note 74.

[125] *Id.*

[126] FED. TRADE COMM'N, *supra* note 14.

IV. OPEN SOURCE SOFTWARE AND OPEN SOURCE LICENSE

Having explained why software should not be patent-eligible, I now analyze open source software and open source licenses identifying pros and cons of open source software and of *open source paradigm*. Open source software seems to be the appropriate model to adopt in the software industry.

In the 1960s and 1970s, the first open source projects were developed in universities and corporate research facilities. Unix, probably the first important open source community product,[127] is an operating system originally developed by AT&T Bell Laboratories and then distributed to the government and academic institutions after U.S. antitrust regulations prohibited AT&T from competing in the computer industry and greatly extending its popularity.[128] In the 1980s, computer users started seeing Unix as a likely universal operating system compatible with all types of computers. Since then, many versions of Unix and other open source software, such as Linux, Apache and Android, have been developed.

If almost everyone has an idea of what closed source software is and how its licensing works, only a few know the meaning of open source software and open source licensing. Why should a firm invest time and money on projects to innovate and develop technology without the possibility of acquiring exclusionary rights? Is the *open source paradigm* economically sustainable in software markets? It is clear why someone should invest in patented inventions like closed software, but the incentives associated with investments in open source are less intuitive. In this Part, I clarify open source licensing and explore the incentives to develop and invest in open source software products.

In the software industry there are two different approaches. The open source approach concerns software creators and scholars that are for a radical decrease of property rights for software. The closed source approach would incentivize the imposition of strong property rights because software has creative commons. The open source community embodied by software developers embraces the first approach.

Software developers hold that contributing to collaborative projects both allows someone else to freely use their software and incentivizes high-quality programs. The Linux kernel, Firefox Web Browser, and Android are only a few examples of open source software. Open source software is opposite to proprietary or closed software and is also called free open source software ("FOSS") to emphasize freedom from control by another and/or ethical issues. Open source software not only means no proprietary software, but also implies that open source code can be reused by anyone in a different project.[129] For example, the Internet browser Mozilla was created with the release of Netscape's source code.[130]

Therefore, open source software is not only free, but its source code is also public. However, open source products are still licensed.

---

[127] Lerner & Tirole, *supra* note 85, at 201.

[128] COLIN RITCHIE, OPERATING SYSTEM INCORPORATING UNIX & WINDOWS 9 (Thomson ed., 4th ed., 2003).

[129] *See, e.g.*, Michal S. Gal, *Viral Open Source: Competition vs. Synergy*, 8 J. COMPETITION L. & ECON. 469, 475 (2012).

[130] Mozilla, *History of Mozilla*, https://www.mozilla.org/en-US/about/history/details/ (last visited Aug. 12, 2015).

A. Open Source Licenses: GPL—LGPL—Creative Commons

A software programmer distributes its software through a license.[131] The license authorizes the licensee to take only those actions within the scope of the granted license. Several types of licenses exist and can be categorized into a proprietary model of copyright (with a restrictive license), and the free/open source model (with an open source license). Generally, firms like Microsoft[132] and Apple adopt a proprietary model to license end-users their products, such as software, for a fee. Conversely, the open source movement opts for the free source model and open licenses, such as GPL, LGPL, BSD (the so-called *FOSS Licenses*), and creative commons licenses to distribute their works.[133]

1. GPL

In 1989, Richard Stallman, the initiator for the open source movement, wrote the first version of the General Public License ("GPL").[134] Open source software, like Firefox, is generally licensed by GPL[135] which allows third parties to retain the right to use, modify, and redistribute an author's copyrighted work. This license implies four freedoms, namely: (i) running the program for any purpose; (ii) studying and modifying the program; (iii) redistributing copies of the software; and (iv) revising the software and releasing the versions to the public.[136] The GPL provides the so-called "copyleft" feature, which implies that any derivative works would have to be released under the GPL.[137] According to the open source movement, the name 'copyright' was changed to 'copyleft' because, in contrast to proprietary software, the GPL of open source software would guarantee freedom.[138]

The logic under open source contrasts with the practice of acquiring monopoly power and maximizing profits. The Seventh Circuit recognized that the GPL "is a cooperative agreement that facilitates production of new derivate works, and agreements that yield new products" [139] otherwise not legally discovered through unilateral action.

---

[131] According to the RESTATEMENT OF PROP. § 512 cmt. a (1944), "the word 'license' is used to describe any permitted unusual freedom of action. It may be used to describe privileges to carry on businesses or to practice callings not otherwise permitted."

[132] However, companies like Microsoft started using also open licenses, such as *Microsoft Community License* and *Microsoft Permissive License*.

[133] *See, e.g.*, Brian Fitzgerald Lero, *The Transformation of Open source Software*, 30 MIS QUATERLY 587, 590 (2006).

[134] *See* Gal, *supra* note 129 at 477.

[135] GNU Operating System, https://www.gnu.org/gnu/gnu.html (last visited Feb. 9, 2015).

[136] Gal, *supra* note 129 at 469.

[137] David Ferrance, *Economic Interests and Jacobsen v. Katzer: Why Open Source Software Deserves Protection under Copyright Law*, 58 J. COPYRIGHT SOC'Y U.S.A. 819, 824 (2011); *see also*, Andrew LaFontaine, *Adventure in Software Licensing: SCO v. IBM and the Future of the Open Source Model*, 4 J. ON TELECOMM. & HIGH TECH. L. 449, 461-63 (2006).

[138] GNU OPERATING SYSTEM, *Licenses*, http://www.gnu.org/licenses/licenses.html#WhatIsCopylefte (last visited Nov. 24, 2015). ("Copyleft guarantees that every user has freedom").

[139] Wallace v. IBM, 467 F.3d 1104, 1107 (7th Cir. 2006).

Although the wording "cooperative agreement" could raise some antitrust concerns, Judge Easterbrook observed open source software and the GPL "have nothing to fear from the antitrust laws."[140] Judge Easterbrook noted that cooperative agreements, like GPL, incentivize creation of new derivative products that "would not rise through unilateral actions" and are lawful.[141] Further, information exchanged through an open source platform between competitors is public.

With respect to GPL's enforceability, two main schools of thought explain how the GPL is enforceable. Richard Stallman, the father of the GPL, argues that the GPL is a non-contractual license.[142] According to Stallman, contract law would imply, among other things, that each distributor should get the user's formal assent to the contract before providing a copy.[143] A different school of thought holds that the GPL is a contract.[144] Professor Mark Patterson observed that the best solution could be "conform[ing] licensing law to clearly defined and already-existing rules based on contract."[145] However, although the Federal Circuit seems to consider a license as a contract,[146] the debate on *license v. contract* of the GPL still exists.

Alleging that the GPL is a pure license and that it is a unilateral grant of rights implies that a software licensor may revoke the GPL grant of rights to third parties at any time for any reason. Embracing the contract interpretation, the GPL might bring damages and infringement lawsuits. The plaintiff might compel the defendant to make her software open source.

As Prof. Eben Moglen observed, if the defendant wrongfully included GPL-licensed code in its own property work it might be "mulcted in damages for the distribution that has already occurred, and prevented from distributing its product further."[147]

However, my purpose is not to develop this aspect but only clarify that the GPL is a standard-form license prepared by Free Software Foundation ("FSF"), and that the issue of the nature of the GPL is still under debate. The peculiarity of the GPL is that if a licensor owns software that incorporates GPL-licensed code, the licensor has to offer such software under the GPL's terms, the so-called "copyleft."[148] However, anyone can use GPL-licensed software and not accept the license. Further, GPL

---

[140] *Id.* at 1108.

[141] *Id.* at 1107.

[142] *See The GPL is a License, Not a Contract, Which is Why the Sky Isn't Falling*, GROKLAW.NET, (Dec. 14, 2003, 9:06 PM), http://www.groklaw.net/article.php?story=20031214210634851.

[143] Richard M. Stallman, *Don't Let 'Intellectual Property' Twist Your Ethos*, GNU OPERATING SYS., (June 9, 2006), http://www.gnu.org/philosophy/no-ip-ethos.html.

[144] *See, e.g.*, Mark R. Patterson, *Must Licenses be Contracts? Consent and Notice in Intellectual Property*, 40 FLA. ST. U. L. REV. 105 (2012). According to Professor Mark Patterson, "[a] requirement that license restrictions be imposed only by contract ensures that intellectual property owners obtain both the consent of licensees to the restrictions and consideration sufficient to make the contract enforceable." *Id.* at 108.

[145] *Id.*

[146] *Id.* at 131, 132. *See e.g.,* Sun Microsystems, Inc. v. Microsoft Corp., 188 F. 3d 1115, 1122 (9th Circuit 1999).

[147] Pamela Jones, *The GPL Is a License, not a Contract* (Dec. 3, 2003), http://lwn.net/Articles/61292/.

[148] Eben Moglen, *Free Software Matters: Enforcing the GPL*, I 2 (Aug. 12, 2001), http://moglen.law.columbia.edu/publications/lu-12.html; *see also* Ferrance, supra note 137, at 824.

covers merely the software. The physical media derived from the software licensed by GPL can be charged.[149]

2. LGPL

The GPL is the most widespread license for open source software,[150] but it is not the only one. FSF introduced a lighter version of GPL, the Lesser GPL ("LGPL"). Similar to the GPL, the LGPL grants the same four freedoms (freedom to run the program, to study how the program works, freedom to improve the program, and to redistribute copies of the software), but it is less restrictive than GPL. LGPL, for example, does not require revealing the source code of the proprietary software. The LGPL is also called 'a weak copyleft license' because code licensed under it could be combined with proprietary code and used in that way. The FSF does not promote licenses encouraging the use of the first and less permissive GPL.[151]

3. Creative Commons

The Creative Commons ("CC") License is a flexible license that allows people to share, use and develop creative works, without worry of infringing copyrights. Through a CC license you can reserve "some rights" instead of "all rights."[152] You can also limit the use of your work to a 'no commercial use.' The GPL license was specifically written for software, whereas the CC license was thought of for any scientific or artistic product. There are a large number of CC licenses available, and hundreds of millions of scientific and academic materials, songs and videos.[153] For example, Wikipedia adopts the CC Attribution Share-Alike license.[154] The idea underlying the CC license was copied from the FSF. At the beginning, the CC license was not compatible with the GPL license; but now the CC license is included in the free software license list.[155] However, in the software context, the CC community

---

[149] *Wallace*, 467 F.3d at 1106.

[150] *See* Sapna Kumar, *Enforcing the GNU GPL*, 1 J. L. TECH. & POL'Y 1, 3 (2006). *See also* Lerner & Tirole, *supra* note 85, at 202.

[151] GNU OPERATING SYSTEM, *Gnu Lesser General Public License,* (Jun. 29, 2007), http://www.gnu.org/licenses/lgpl-3.0.en.html; *see also* GNU OPERATING SYSTEM, *Why you Shouldn't Use the Lesser GPL for your Next Library,* http://www.gnu.org/licenses/why-not-lgpl.en.html; Eli Greenbaum, *Open Source Semiconductor Core Licensing*, 25 HARV. J. LAW & TEC 131, 140 (2011).

[152] CREATIVE COMMONS, *ABOUT*, http://creativecommons.org/about; *see also,* Jacobsen v. Katzer, 535 F.3d 1373, 1378 (Fed. Cir. 2008).

[153] *Id.*

[154] CREATIVE COMMONS, *Wikipedia + CC BY-SA = Free Culture Win!*, http://creativecommons.org/weblog/entry/15411. ("Wikipedia community and Wikimedia Foundation board approved the adoption of the Creative Commons Attribution-Share CC BY-SA) license as the main content license for Wikipedia and other Wikimedia sites.").

[155] CREATIVE COMMONS, *GPL*, http://creativecommons.org/tag/gpl.

incentivizes the use of the GPL license recommending "against the use of creative commons license for software."[156]

B. Motivations for Contributing to Open Source Software

The first economic dogma is based on preserving economically efficient incentives. Where are these incentives in open source paradigm located? If no individual or firm has proprietary control of the open source software, what economic incentives do those who work for developing and improving that software have?

Open source has the recognized capacity to produce higher quality products[157] with more innovative character[158] than similar proprietary products (Android is an example). A collaborative and decentralized development model provides better and more rapid solutions than a centralized and static enterprise model. By doing so, open source promoters argue that this flexible and open source paradigm benefits innovation increasing consumer welfare. However, the key inquiries of what economic incentives firms and individuals have in investing in open source projects remains unknown. The logic of patent law is that by providing the right to exclude others from making, using, offering for sale, or selling its own product through a patent, the legislator preserves the economic incentives of those who want to invest in research and development. Microsoft, for example, has one of the most relevant patent portfolios in the world and has not shown any signs of slowing such activity over the years.[159]

Conversely, open source products are inherently free. As scholars like Jean Tirole and Ronald J. Mann observed, one of the goals of open source projects is to contrast and decrease the monopoly power of companies like Microsoft or Apple.[160] It sounds like an economic strategy to compete with technological giants otherwise immune to the competitive game. While Microsoft or Apple usually develop a vertical integration strategy, the open source model, as Judge Easterbrook noted, calls to mind the economics of joint venture.[161] Similar to open source strategy, the latter relies on cooperation amongst different people or groups.

Another reason to invest in an open source project is to create and develop a product that serves firms to develop and profit from complementary and proprietary products.[162] The major investors of Linux software are IBM, Intel, HP, Fujitsu,

---

[156] CREATIVE COMMONS, *Frequently Asked Questions*, https://wiki.creativecommons.org/wiki/Frequently_Asked_Questions#Can_I_use_a_Creative_Commons_license_for_software.3F.

[157] *See, e.g.*, Lero, *supra* note 133, at 587.

[158] Open Source Initiative, *About the Open Source Initiative* (last visited Feb. 9, 2015), http://opensource.org/about. *See also* Gal, *supra* note 129, at 475.

[159] Steve Branchman, *Microsoft Patents Business Data Services, Anti-Phishing Scanners and Tailored Web Services*,IPWATCHDOG, http://www.ipwatchdog.com/2014/10/04/microsoft-patents-anti-phishing/id=51500/.

[160] Lerner & Tirole, *supra* note 85, at 225; Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?*, 20 HARV. J.L. & TECH. 1, 23 (2007).

[161] *Wallace*, 467 F.3d at 1108.

[162] Mark A. Lemley & Ziv Shafir, *Who Chooses Open-Source Software?*, 78 U. CHI. L. REV. 139, 140 (2011).

Novell, Red Hat, and General Motors.[163] Each one of these companies is active in a different product market supplementary to Linux. For example, IBM[164] invested a lot in distributing GNU/Linux and other free software communities' products to both increase marketability of its hardware products and decrease the cost of its services.[165] Economic incentives to invest in open source projects, such as Linux, seem linked to the value chain concept.

As Prof. Ronald J. Mann noted, "investing in Linux is a rational step for the individual members of the OSDL not because it might harm Microsoft or generate profits from direct sales, but because developing Linux as a high-quality operating system permits each of them to develop complementary goods and services in their respective core competencies."[166] It means that Linux or the free web can be compared with platforms[167] used by third-party software to develop complementary goods.

The creation of a common platform to develop such products bears to mind the essential facility doctrine, namely, the conjecture that forced sharing of essential inputs at regulated rates increases competition and fosters innovation.[168] Historically, the essential facility doctrine was successfully applied in the telecommunication industry. However, in contrast to this doctrine, the common platform created by open source community is not a forced sharing of essential information. Rather, it is a voluntary platform that such community created for exchanging and sharing knowledge and information among computer users to identify and develop even more advanced software.

In sum, big companies like Google, Samsung and IBM are important investors in the open source community, but the protagonists of that community are people moved by intellectual curiosity like Linus Torvalds or Tim Berners-Lee.[169] Together,

---

[163] *See, e.g.*, Lerner & Tirole, *supra* note 85, at 198.

[164] Further, IBM, although its Eclipse IDE was valued at 40 million of dollars, decided to move it to open source, increasing its popularity and the market for its ancillary products. *See* Lero, *supra* note 133, at 592.

[165] Eben Moglen, *Free Software Matters: Free Government 2* (Sep. 14, 2002), http://moglen.law.columbia.edu/publications/lu-23.pdf. *See also*, Lero, *supra* note 133, at 592. ("Several companies leverage open source as a base upon which they offer products other than software.")

[166] Mann, *supra* note 160 at 25.

[167] *See* Lero, *supra* note 133, at 589. The most relevant FOSS products, such as Apache, Linux and Mozilla "are all example of horizontal infrastructure software."

[168] MCI Commc'n Corp. v. American Tel. & Tel. Co., 708 F.2d 1081, 1132 (7th Cir. 1983). Here,

> [t]he jury found that AT&T unlawfully refused to interconnect MCI with the local distribution facilities of Bell operating companies—an act which prevented MCI from offering FX and CCSA services to its customers. A monopolist's refusal to deal under these circumstances is governed by the so-called *essential facilities doctrine*. Such a refusal may be unlawful because a monopolist's control of an essential facility (sometimes called a "bottleneck") can extend monopoly power from one stage of production to another, and from one market into another. Thus, the antitrust laws have imposed on firms controlling an essential facility the obligation to make the facility available on non-discriminatory terms.

Id. at 1132 (emphasis added).

[169] According to Linus Torvalds, "I really don't think you need all that much 'quid pro quo' in programming—most of the good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program." *See FM Interview with Linus Torvalds: What Motivates Free Software Developers?*, 3 FIRST MONDAY, (Mar. 1998), *available at*

all, for different reasons, today contribute in the development of a free and open technological innovation process.

C. Vertical Integration v. Open Source Licensing – Android Case

Android shows that the open source paradigm can be profitable and successfully implemented in the technology industry. According to the smartphone research firm Strategy Analytics, Android accounted for 81.3 percent of worldwide smart phone shipments in the third quarter of 2013.[170] But how does open source paradigm diverge from a vertical integration strategy?

Software markets promote closed operating systems (*see* Apple's iOS) as well as open operating systems (*see* Android). While closed operating systems are vertically integrated—Apple produces Apple's iOS for its own iPhones—open operating systems are common platform (unphysical infrastructure) shared and used by several competing corporations. Open source products are the result of a sort of joint venture among competitors to develop a free, open, and competitive platform, which serves to create more advanced technological complementary goods. Companies that invest in open source products do not profit from the open source product, but from complementary goods based on the open source technology.

The open source model threatens the closed source model in the software market. For example, significant network effects characterize the technology industry and software market. This means that if the majority of consumers buy Apple iOS products, the value of Apple's operating system increases. Conversely, if the majority of consumers buy smartphones with Android, the value of Android's smartphone increases. In industries characterized by network effects, open source provides unique procompetitive advantages. In the context of standards, for example, if the standard is proprietary, the costs of switching to a new standard are high. Although better alternatives are developed, users are hooked in and the standard holder can extract super competitive rents. This phenomenon (the higher royalty based on switching costs) is well known as *hold-up* value of the patent and consumers bear the cost of reduced innovation and higher prices of the standard. Conversely, if the standard is open, although it is dominant, a monopolist extracting rent cannot exist and no such hold-up is possible because of the cost of switching.[171] Open source paradigm guarantees transparency, and transparency implies low-cost and no-cost versions of the good that can be offered. Thus, open source standard means better allocation of productive resources; open source products, being some of the most advanced products offered for free, create pro-competitive and pro-consumer

---

http://www.firstmonday.org/issues3_3/torvalds/index.html (on file with the University of Illinois Law Review); *see also* Lemley & Shafir, *supra* note 162, at 140.

[170] Kukil Bora, *Android Powers 81 Percent Of All Smartphones Shipped Worldwide In Q3 As iOS Market Share Drops*, INT'L BUS. TIMES, (Nov. 2, 2013) http://www.ibtimes.com/android-powers-81-percent-all-smartphones-shipped-worldwide-q3-ios-market-share-drops-1451134. In October 2015, Android held a market share of 52.61 percent of worldwide Mobile/Tablet Operating System Market and Apple IOS got 40.26 percent of market share. *Mobile/Tablet Operating System Market, NETMARKETSHARE,* (October, 2015), https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1.

[171] Boyle, *supra* note 75, at 31.

advantages.[172] Prof. Moglen also emphasizes that monopoly power is weakened by the adoption of free software.[173] The open source paradigm, however, might easily infringe patents and standards, questioning the entire patent system in the technological world.

The crucial question at hand is: Do Android and other open source software represent the best model in terms of competition and firms' economic incentives to innovate? Berners-Lee recognized that "commercial applications including eBay, Google, Yahoo, and Amazon.com are but a few examples of the extraordinary innovation that is possible because of the[ir] open, standards-based, royalty-free" nature.[174]

Thus, for these and the other reasons explained in this article, I agree with Tim Berners-Lee, and strongly believe that the open source model is the most efficient economic model that could be implemented in the software industry. The implementation of the open source model thus far, and its benefits to consumers, implies that we should adopt it in many other technological and innovative industries. Nowadays, open source appears as the new business model created/developed by the free and open movement of the Internet. The same Internet has caused the third industrial revolution; business models developed before such a revolution are hardly efficient or competitive now.

V. COPYRIGHT AND OPEN SOURCE

Last but not least, even without software patents, presumably software could be copyrighted. Thus, a legitimate question would be: Is the copyright protection of software the sought-after compromise between patent protection and no protection at all?

Before answering this question, I explain what copyright software implies and the main differences between software copyright and software patent.

Further, I explain how copyright and open source can coexist identifying the best solution in terms of licenses to guarantee the adequate protection for software developers and technological innovation as a whole.

A. Patent and Copyright Protection in Comparison

A legal system usually provides rights and protections for intellectual properties, among them are patent and copyright protection. A patent protects an invention of a

---

[172] *Id.* at 62.

[173] Eben Moglen, *Free Software Matters: Free Government 1* (Sep. 14, 2002), *available at* http://moglen.law.columbia.edu/publications/lu-23.pdf.

[174] *Testimony of Sir Timothy Berners-Lee CSAIL Decentralized Information Group Massachusetts Institute of Technology, Hearing on the Digital Future of the United States: Part I—The Future of the World Wide Web* (2007), http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.html.

tangible thing that is new, useful and non-obvious.[175] Conversely, copyrights protect creative works in art, literature, theatre, photography, film & TV, and programming.

Patent protection gives the inventor, for a limited period of time, the exclusive rights to make, use, and sell the invention in the State where the patent was issued. Similarly, the author of a copyrighted product has the only right to reproduce, distribute, and perform the copyrighted work.[176] Thus, how much of a difference does copyright protection make? Here are some thoughts.

1. Copyright Protection v. Patent Protection

Despite patents and copyrights granting their respective owners similar exclusive rights, patents cover inventions, such as operational methods and procedures, whereas copyrights concern expressions and creative works. Copyrights do not protect the core idea or facts upon which the author's creative expression is based. The discovered historical or scientific facts and ideas are in the public domain, and anyone is free to use such facts or ideas.[177] Rather than only protecting the subject matter of paintings or writings, copyrights cover forms of expression. Further, some copyright exceptions exist. In certain circumstances you can invoke the 'fair use' rule, which allows (without asking an author's permission) a limited use of copyrighted works.[178] For example, the fair use rule allows researchers to quote short passages in a scientific, academic, or technical work to clarify author's opinions or observations. The fair use privilege likely represents the most important limitation on copyright holders' rights.

2. Patent and Copyright in the Software Context

In the context of software, copyright protection works the same way as it works in any other artistic/literary work. Software copyright implies that a program developer owns the right of copying and adapting his own program's code, but anyone can reprogram the copyrighted software without infringing the copyright.

As Sam Williams and Richard Stallman observed, the fact that copyrights do not cover ideas means that "a developer is free, under copyright, to implement in his own

---

[175] *See* U.S. Patent Act, 35 U.S.C. §§ 101 *et. seq*. In the United States there are several types of patents, among which are utility patents and design patents. U.S. PAT. & TRADEMARK OFFICE, TYPES OF PATENTS, http://www.uspto.gov/web/offices/ac/ido/oeip/taf/patdesc.htm (last visited Nov. 23, 2015).

[176] The U.S. Copyright Act (17 USC §§ 100 *et. seq*.) establishes copyright protection "for original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device."

[177] *See, e.g.,* Steve Posner, *Can a Computer Language be Copyrighted? The State of Confusion in Computer Copyright Law*, 11 COMPUTER L. J. 97, 104 (1992).

[178] Nolo Law for All, *The 'Fair Use' Rule: When Use of Copyrighted Material is Acceptable*, http://www.nolo.com/legal-encyclopedia/fair-use-rule-copyright-material-30100.html.

code features and commands he has seen in existing programs."[179] Thus, "[i]t is likewise lawful—through hard work—to decode how a binary program works, and then implement the same ideas and algorithms in different code. This practice is known as "reverse engineering."[180]

In the United States, software benefits from both patent and copyright protection. Patenting software, a set of instructions by which programmers tell computers what to do, means to patent a process, a specific human order translated into machine language. By doing so, the patent covers the idea and the facts underlying the software, and it is difficult to reproduce similar elements/processes or facts without infringing the related patents.

Sam Williams and Richard Stallman noted that patent holders have the power to prohibit the use of the patent's underlying idea to an independent developer of software programs.[181] In contrast to copyrighting of software, if the software is patented, a programmer cannot implement the patented invention and algorithms in different code.

As scholars recognize, patenting software stifles the creation of even more developed and efficient software, and compromises the development of technology.[182] Protecting software developers and recognizing their work is fundamental, but an inappropriate protection can limit innovation instead of incentivizing technological development.

3. Software Copyright—Jacobsen Case

The *Jacobsen* case helps understand how copyrights and open source software can coexist and why copyright represents the appropriate legal protection for software. Jacobsen licensed its computer programming through an Artistic License, an open source copyright license.[183] Jacobsen sued Katzer for copying and modifying its software, going beyond the scope of the Artistic License.

Jacobsen handled the Java Model Railroad Interface ("JMRI"), an open source software group which created, with other participants, the DecoderPro computer programming application. Katzer started offering competing software, the Decoder Commander that used the DecoderPro software files without including the terms of the Artistic License.[184]

---

[179] SAM WILLIAMS, FREE AS IN FREEDOM (2.0): RICHARD STALLMAN AND THE FREE SOFTWARE REVOLUTION 112 (Richard M. Stallman ed., 2d ed. 2010), *available at* https://sagitter.fedorapeople.org/faif-2.0.pdf.

[180] *Id.*

[181] *Id.* at 113.

[182] Michele Boldrin & David K. Levine, *The Case Against Patents*, FED. RES. BANK OF ST. LOUIS, RES. DIV. (2012), http://research.stlouisfed.org/wp/2012/2012-035.pdf.

[183] *Jacobsen*, 535 F. 3d at 1376.

[184] *Id.* at 1377. In particular, "the Decoder Commander software did not include (1) the author's name, (2) JMRI copying notices, (3) references to the Copying file, (4) an identification of SourceForge or JMRI as the original source of the definition files, and (5) a description of how the files of computer code had been charged from the original source code." *Id.*

Copyrighted works are usually distributed for money. Open source licenses do not ask for money, but there are several benefits (such as economic benefits) under the open source license.[185]

The crucial issue discussed in *Jacobsen*'s appeal was whether to regard the terms of the DecoderPro license as conditions of the copyright license or covenants. If such terms were conditions, or both covenants and conditions, then they would concern copyright law and limit the scope of the license.[186] Conversely, if the Artistic License's terms were merely covenants, Jacobsen could have invoked only a contract law violation. In contrast to the District Court's holding,[187] the Court of Appeals recognized that the Artistic License "use[d] the traditional language of conditions by noting that the right to copy, modify, and distribute are granted 'provided that' typically denotes a condition."[188]

Copyright owners who opt for open source licensing control the distribution and modification of the copyrighted work. Money damages are not necessary to support the copyright right to exclude.[189] Therefore, the Court of Appeals stated that Katzer infringed Jacobsen's copyright by going "outside the scope of the Artistic License to modify and distribute the copyrighted materials without copyright notices and a tracking of modifications from the original computer files."[190]

In sum, Jacobsen's case clarified that open source software is entitled to the same legal protection as proprietary software.[191] On the one hand, open source licensing means that the copyrighted work is available to the public for free, and on the other hand the copyright holder keeps the right to decide by whom and how its work can be used, modified and sold.

VI. FINAL CONSIDERATIONS—WHY OPEN SOURCE PARADIGM?

This article started with the software patentability debate, explaining how the granting of software patents is harmful for technological innovation, and should not be allowed. Software patentability limits the open source community from freely developing software and offering new technologies to consumers for free. By continuing to patent software, Patent and Trademark offices, as well as courts that uphold those patents, appear to limit competition in the affected markets and retard the crucial innovation of technology—a trend that is constantly reinforced by legislators, regulators, and the legal community. Open source software is the natural response to software development and the Internet; the open Web's success is clear evidence that the open source paradigm works in technological industries, allowing everybody to develop even better complementary products based on the open source

---

[185] *Id.* at 1379 ("The Eleventh Circuit has recognized the economic motives inherent in public licenses, even when profit is not immediate.")

[186] *Id.* at 1380.

[187] *Id.* at 1381; *see also*, Victoria Nemiah, *License and Registration, Please: Using Copyright "Conditions" to Protect Free/Open Source Software*, 3 NYU J. INTELL. PROP. & ENT. L. 358, 378-79 (2014).

[188] *Id.* at 1381.

[189] *Id.* at 1382.

[190] *Id.*

[191] Ferrance, *supra* note 137.

platform. The open source model promotes the free sharing of information, encouraging anyone to use and develop the open source technology. Moreover, even without software patents, software can be copyrighted. Software copyrights provide adequate protection for software developers, recognizing the author's creation and providing legal protection from abusive conduct.

Open source products are offered free of charge, so such products will inevitably increase consumer welfare. Thanks to the open source paradigm, consumers benefit from high-quality products, such as Linux, free of charge.

Furthermore, the software industry is not the only industry that would benefit from a widespread adoption of the open source model.[192] The open source paradigm could be the most appropriate economic model for a number of technological markets. The open source paradigm has nothing to fear from either the antitrust laws or intellectual property laws. By offering products free of charge, open source goods increase consumer welfare, attaining the main goal of antitrust regulations. Encouraging the development of advanced products is exactly the purpose of patent law.

Open source implies transparency, democracy, freedom, cooperation, development, efficiency, innovation, creative, and equality. Therefore, open source can be better expressed as an economic/social paradigm that is able to perform transparency, democracy, freedom, cooperation, development, efficiency, innovation, creative, equality in software and other crucial markets.

---

[192] Mark A. Lemley & Ziv Shafir, *Who Chooses Open-Source Software?* 140 (Stan. L. & Econ. Olin Working Paper No. 382, 2009), *available at* http://ssrn.com/abstract=1495982; *see also* Andrew W. Torrance, *Open Source Human Evolution*, 30 WASH U. J.L. & POL'Y 93, 125-28 (2009).