Santa Clara University School of Law

Santa Clara High Technology Law Journal

Volume 18 | Issue 2 Article 3

January 2002

Source Code versus Object Code: Patent Implications for the Open Source Community

Daniel S. Lin

Matthew Sag

Ronald S. Laurie

SOURCE CODE VERSUS OBJECT CODE: PATENT IMPLICATIONS FOR THE OPEN SOURCE COMMUNITY

Daniel Lin,† Matthew Sag,†† and Ronald S. Laurie†††

I. INTRODUCTION

Since the Federal Circuit's 1995 decision in *In re Beauregard* and the United States Patent and Trademark Office's ("PTO") subsequent issuance of its Guidelines for Computer Related Inventions ("PTO Guidelines") in 1996, computer programs embodied in a computer-readable medium are now considered patentable subject matter under 35 U.S.C. § 101 by the PTO.[1] Specifically, patent claims, now commonly referred to as "Beauregard claims," that recite an invention embodied in a computer-readable medium are readily allowed by the PTO as long as they satisfy the novelty, non-obviousness, and utility requirements of 35 U.S.C. §§ 102 and 103.[2] However, the Federal Circuit has never definitely concluded whether such embodied computer programs are indeed

† Daniel Lin is an associate at Skadden, Arps, Slate, Meagher & Flom LLP in Palo Alto, CA. He can be reached at dlin@skadden.com.

†† Matthew Sag is an associate at Skadden, Arps, Slate, Meagher & Flom LLP in Palo Alto, CA. He can be reached at msag@skadden.com.

††† Ron Laurie is a partner and the head of the IP Strategies and Transactions Practice at Skadden, Arps, Slate, Meagher & Flom LLP in Palo Alto, CA. He can be reached at rlaurie@skadden.com. The authors would like to thank David Hansen, Fred Kim, Gene Su, Joseph Yang, and participants at *Information Insecurity: Protecting Data in the Digital Age* at Santa Clara University for their helpful discussions and comments.

1. *In re* Beauregard, 53 F.3d 1583 (Fed. Cir. 1995) ("The Commissioner now states 'that computer programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 . . . .'"); Examination Guidelines for Computer Related Inventions, 61 Fed. Reg. 7478, 7481 (Patent & Trademark Office, U.S. Dep't of Commerce) (Feb. 28, 1996) [hereinafter Examination Guidelines] ("When functional descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases."). *See also* 35 U.S.C. § 101 (2000).

2. 35 U.S.C. §§ 102–103.

patentable. Therefore, the question is raised, what does the PTO mean by a "computer program?"[3]

To appreciate the ambiguity of the term "computer program," imagine a scenario in which a programmer at a security software company is searching the Web for an elegant solution to a cryptographic problem. He comes across a cryptography open source project Web site that appears to offer such a solution. The programmer downloads the source code from the Web site onto his computer's hard drive. However, after inspecting the source code, he concludes that the solution provided by the source code is not sufficiently robust to be used at his company and decides *not* to use the code. Now, further assume that unknown to either the programmer or the open source project, the functionality described in the downloaded source code is covered by a third party's patent (i.e., in Beauregard form). By simply downloading the source code onto his hard drive (i.e., a computer-readable medium), has the programmer infringed the third party's patent?

In a world where source code on a hard drive *is* a computer program embodied in a computer-readable medium the programmer *has* infringed the third party's patent, because by merely downloading the source code, the programmer has "made" the computer program under the Patent Act.[4] Thus, under such an interpretation of "computer program," any person or company wishing to assess the quality of source code by downloading a copy simply to examine it, without even compiling or executing it, could potentially be infringing another's patent. Such potential for patent liability could discourage the widespread distribution of source code that produces the exchange of new ideas, innovative theories and techniques, and secure coding practices that are so valued by the open source ideal. As such, those in the open source community typically view "software patents" as "the monster hiding under every software

3. The PTO Manual of Patent Examining Procedure § 2106 defines a computer program as "a set of instructions *capable of being executed* by a computer." PATENT & TRADEMARK OFFICE, U.S. DEP'T OF COMMERCE, MANUAL OF PATENT EXAMINING PROCEDURE § 2106, at 2100-13 (8th ed. Aug. 2001) (emphasis added) [hereinafter MPEP], *available at* http://www.uspto.gov/web/offices/pac/mpep/mpep.htm.

4. 35 U.S.C. § 271 (2000) ("Except as otherwise provided in this title, whoever, without authority *makes*, uses, offers to sell or sells any patented invention, within the United States or imports into the United States any patented invention during the term of the patent therefor, infringes the patent) (emphasis added). *Id.* Downloading source code creates a local copy of that source code, thereby effectively "making" the source code.

developer's bed."[5] Nevertheless, rather than addressing the ambiguities of computer software patentability in the current legal framework, much of the open source discussion regarding patents focuses on the lack of novelty or obviousness in software patent claims.[6]

It is far from clear whether we live in a world where source code on a hard drive (or any other computer-readable medium) is considered statutory subject matter as a "computer program" by the PTO or the Federal Circuit. This Article explores the current legal framework regarding computer software and patents. It explores the distinctions between source code and object code and discusses the legal ramifications of these distinctions in patent law. Part II provides a brief discussion of the technical distinctions between source code and object code. Part III explores the issue of whether source code infringes software patents, presents an argument that the infringement of software by source code may overextend patent jurisprudence, and points out the ambiguities of the PTO with regard to Beauregard claims when applied to source code. Finally, Part IV examines the implications of the foregoing for the open source community and concludes that if source code does not infringe patents, then many important open source activities may be free from software patent concerns.

---

5. DONALD K. ROSENBERG, OPEN SOURCE: THE UNAUTHORIZED WHITE PAPERS 240 (2000). *See also* RUSSELL C. PAVLICEK, EMBRACING INSANITY: OPEN SOURCE SOFTWARE DEVELOPMENT 161 (2000) ("The use of software patents has been a real problem in the Open Source world."); Richard Stallman, *The GNU Operating System and the Free Software Movement, in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION 53–70 (Chris DiBona et al. eds., 1999) ("The worst threat we face comes from software patents, which can put algorithms and features off limits to free software for up to twenty years."), *available at* http://www.gnu.org/gnu/thegnuproject.html (last visited Apr. 11, 2002).

6. *See, e.g.*, Richard Stallman, *The Anatomy of a Trivial Patent*, LINUX TODAY, May 26, 2000 ("Programmers are well aware that many of the software patents cover laughably obvious ideas."), *at* http://linuxtoday.com/news_story.php3?ltsn=2000-05-26-004-04-OP-LF; LEAGUE FOR PROGRAMMING FREEDOM, AGAINST SOFTWARE PATENTS, Feb. 28, 1991, *at* http://lpf.ai.mit.edu/Patents/against-software-patents.html (last modified Apr. 29, 1994); Lawrence Lessig, *The Problem with Patents*, THE INDUSTRY STANDARD, Apr. 23, 1999 ("What is 'novel,' 'nonobvious' or 'useful' is hard enough to know in a relatively stable field. In a transforming market [such as the Internet], it's nearly impossible for anyone . . . to identify what's 'novel.'" (alteration added)), *at* http://www.thestandard.com/article/display/0,1151,4296,00.html. Additionally, two Web sites provide more information on the case against software patents: League for Programming Freedom, *at* http://lpf.ai.mit.edu/Patents/patents.html and Free Patents: Protecting Innovation and Competition in the IT Industry, *at* http://www.freepatents.org/.

II. SOURCE CODE VERSUS OBJECT CODE

Source code has been described as a computer program written in a high level human readable language.[7] In contrast, the related object code is the same computer program written in computer readable format, which is required for the program's execution by a computer.[8] One important difference between source code and object code is that source code is generally platform-independent, meaning that it does not refer to the intricacies of any particular type of computer.[9] In contrast, object code is platform-specific and must necessarily refer to the inner workings of the particular computer (e.g., memory locations, instruction sets, etc.) upon which the object code is to be executed.[10] In order to convert source code into object code, the source code is provided to a compiler, a separate computer program that reads the source code and translates it into the object code.[11] As the compiler is executed, it performs lexical, syntactic,

---

7. Reiffen v. Microsoft Corp., 214 F.3d 1342, 1344 (Fed. Cir. 2000) ("A source program is a computer program written in a high level human readable language which the application refers to as source code; the end product of the compilation of the source program is a binary machine language composition which the application refers to as object code, and which is required for the program's execution by a computer.").

8. *Id.*

9. Webopedia defines a "high-level language" as "[a] programming language such as C, FORTRAN, or Pascal that enables a programmer to write programs that are *more or less independent* of a particular type of computer." WEBOPEDIA, HIGH-LEVEL LANGUAGE (emphasis added), *at* http://www.webopedia.com/TERM/h/high_level_language.html (last modified Oct. 26,1996).

10. Webopedia defines "machine languages" as "the *only* languages understood by computers." WEBOPEDIA, MACHINE LANGUAGE (emphasis added), *at* http://www.webopedia.com/TERM/m/machine_language.html (last modified Nov. 16, 2001). For the purposes of this Article, object code is synonymous with "machine language" or "executable code."

11. Object code may be defined in various ways. For the purposes of this Article, the term "object code" will be synonymous with "machine language" or "executable code," see *supra* note 10. Webopedia defines object code as:

> The code produced by a compiler. Programmers write programs in a form called source code. The source code consists of instructions in a particular language, like C or FORTRAN. Computers, however, can only execute instructions written in a low-level language called machine language.
>
> To get from source code to machine language, the programs must be transformed by a compiler. The compiler produces an intermediary form called object code. *Object code is often the same as or similar to a computer's machine language.* The final step in producing an executable program is to transform the object code into machine language, if it is not already in this form. This can be done by a number of different types of programs, called assemblers, binders, linkers, and loaders.

WEBOPEDIA, OBJECT CODE (emphasis added), *at* http://www.webopedia.com/TERM/o/object_code.html (last modified Sept. 1, 1996).

and semantic analyses of the source code, which is stored in a source buffer in the computer's memory, outputting the compiled object code into an object buffer.[12]

During the compilation process, the compiler can significantly improve the performance of the object code (a process known as "optimization"), by adjusting and manipulating code generation in certain ways. For example, a compiler can optimize the object code by improving the efficiency of loops, procedure calls, address calculations, and peephole transformations.[13] Such improvements are known as machine-independent optimizations since they do not take into consideration any properties of the computer that will execute the object code.[14] Furthermore, in order to generate object code, a compiler must have precise knowledge of the instruction set of the computer upon which the code is to be executed and therefore can create further efficiencies through machine-dependent optimizations such as register allocation and the utilization of special machine-instruction sequences.[15] Depending upon the skill and objectives of the compiler writer, there is great variety in the level of code optimization that different compilers perform, resulting in significantly different object code given a particular piece of source code.[16]

III. CAN SOURCE CODE INFRINGE PATENT CLAIMS?

As specifically enumerated by the Patent Act, only a process, machine, manufacture, or composition of matter can be patented.[17] These four express statutory categories (known as "statutory subject matter") exhaust the possible subject matter that can be patentable

---

Furthermore, "object code" in this Article means "absolute machine language" that can be placed in a fixed location in a computer's memory and immediately executed. In contrast, a "relocatable machine language program," also known as an object module, allows subprograms to be compiled separately and linked together and laded for execution by a link loader. *See* ALFRED V. AHO ET AL., COMPILERS: PRINCIPLES, TECHNIQUES, AND TOOLS 514 (1985).

12. *See* AHO, *supra* note 11, at 587.

13. *See id.*

14. *See id.* at 585.

15. *See id.* at 587.

16. *See id.*

17. 35 U.S.C. § 101 (2000) ("Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.").

inventions.[18] However, the Supreme Court has given a broad interpretation to these categories, indicating that "Congress intended statutory subject matter to 'include anything under the sun that is made by man.'"[19] In the computer software arts, only three of the four express statutory categories are implicated. These are process, machine, and manufacture claims. This section concludes that only object code can be implicated in process and machine claims. Furthermore, while object code that is embodied on a computer-readable medium infringes a manufacture claim (i.e., *Beauregard* claim), it is unclear whether source code that is similarly embodied also constitutes statutory subject matter that infringes such claims.

A. Computer Software Claimed as a Machine

The Federal Circuit's jurisprudence regarding machine claims in the computer software arts culminated in its 1998 landmark decision in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*[20] In *State Street*, the Federal Circuit developed a new "practical utility" test to determine whether a machine claim related to software was statutory subject matter by simply assessing whether the software produced "a useful, concrete, and tangible result."[21] The claim at issue in *State Street* was directed to a "data processing system," which the court construed as a machine claim, which is proper statutory subject matter under § 101.[22] In particular, the claim included means-plus-function elements, for which the related structures disclosed in the specification were arithmetic logic circuits configured to perform certain tasks.[23] As described in the patent specification, such "configurations" to the arithmetic logic circuits were effected by software.[24] In order to effect such arithmetic logic circuits, such

18. *See* Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 483 (1974) ("[N]o patent is available for a discovery, however useful, novel, and nonobvious, unless it falls within one of the express categories of patentable subject matter of 35 U.S.C. § 101.").

19. Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980); *see also* Diamond v. Diehr, 450 U.S. 175, 182 (1981); Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 154 (1989); *In re* Alappat, 33 F.3d 1526, 1542 (Fed. Cir. 1994) ("Thus, it is improper to read into § 101 limitations as to the subject matter that may be patented where the legislative history does not indicate that Congress clearly intended such limitations.").

20. State Street Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368 (Fed. Cir. 1998).

21. *Id.* at 1373.

22. *Id.* at 1372.

23. *Id.* at 1371–72.

24. U.S. Patent No. 5,193,056 (issued Mar. 9, 1993) ("The portfolio/fund accountant makes use of a personal computer 44 programmed with software 50.").

software must necessarily be object code, not source code, that is loaded into the memory of a personal computer.

The decision in *State Street* followed a string of cases in the mid 1990s that provided the Federal Circuit's rationale for favoring the patenting of general purpose computers running software (necessarily in object code form, not source code).[25] Most significantly, in *In re Alappat*, the court held that a general-purpose computer programmed to perform particular functions pursuant to instructions from software effectively created a new machine that could be patentable under § 101.[26] Therefore, under current jurisprudence, software, as embodied within a general-purpose computer, is patentable as a machine that realizes the functionality of the software, as long as such software produces "a useful, concrete, and tangible result."[27] Necessarily, such software must be object code rather than source code.

B. Computer Software Claimed as a Process

One year after *State Street*, in *AT&T Corp. v. Excel Communications, Inc.*, the Federal Circuit extended its *State Street* decision to process claims.[28] The *AT&T* court held that a process claim need not physically transform the subject matter of the invention from one form to another.[29] Rather, the inquiry is whether the mathematical algorithm used in the method is applied in a practical manner to produce a useful result.[30] As such, the Federal Circuit essentially recognized that software-related processes were no

---

25. *See In re* Beauregard, 53 F.3d 1583, 1584 (Fed. Cir. 1995) ("'[C]omputer programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 and must be examined under 35 U.S.C. §§ 102 and 103.'"); *In re* Alappat, 33 F.3d 1526, 1545 (Fed. Cir. 1994) ("[A] computer operating pursuant to software *may* represent patentable subject matter, provided, of course, that the claimed subject matter meets all of the other requirements of Title 35."); *In re* Lowry, 32 F.3d 1579, 1583–84 (Fed. Cir. 1994) (Particular data structures are statutory subject matter because, "more than mere abstraction, . . . data structures are specific electrical or magnetic structural elements in a memory . . . that provide increased efficiency in computer operation."); *In re* Warmerdam, 33 F.3d 1354, 1361 n.6 (Fed. Cir. 1994) ("[T]he storage of data in a memory physically alters the memory, and thus in some sense gives rise to a new memory."); Arrythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053, 1060 (Fed. Cir. 1992) (Computer-performed operations that simply "transform a particular input signal to a different output signal, in accordance with the internal structure of the computer as configured by electronic instructions," are statutory subject matter.). *See also* Examination Guidelines, *supra* note 1, at 7479.

26. *In re* Alappat, 33 F.3d at 1545.

27. State Street Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 1373 (Fed. Cir. 1998).

28. AT&T Corp. v. Excel Communications, Inc., 172 F.3d 1352, 1358 (Fed. Cir. 1999).

29. *Id.*

30. *Id.* at 1360.

different from other inventions with regard to using principles of novelty, non-obviousness and utility to determine patentability. Consequently, software can be patented as a process claim whose elements describe the functionality of the software, as long as such software produces "a useful, concrete, and tangible result."[31] It is important to note, however, that such software process claims can only be infringed when the process is practiced—that is, when the software is actually running on the computer—and for the software to be running on the computer, that software must be in object code, not source code.

C. Computer Software Claimed as a Manufacture

Since the Federal Circuit's 1995 decision in *In re Beauregard* and the issuance of the PTO Guidelines in 1996, the PTO has been readily allowing computer programs embodied in a computer-readable medium as proper manufacture claims under 35 U.S.C. § 101.[32] Nevertheless, the Federal Circuit has never definitively decided the issue. In *Beauregard*, the PTO Board of Patent Appeals and Interferences' ("Board") rejected Beauregard's computer program product claim on the basis of the printed matter doctrine.[33] However, during the appeal's pendency, apparently on the heels of the Federal Circuit's decision in *In re Lowry* on August 26, 1994, the PTO, reversing its previous position, stated that "computer programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 and must be examined under 35 U.S.C. §§ 102 and 103."[34] Because the Commissioner ultimately agreed with *Beauregard* that the printed matter doctrine was not applicable, no case or controversy existed and the Federal Circuit vacated the Board of Patent Appeals and Interferences' rejection.[35] Therefore, since *Beauregard*, the PTO has accepted such "computer-readable medium" claims, commonly

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

32. *In re* Beauregard, 53 F.3d 1583,1584 (Fed. Cir. 1995) ("The Commissioner now states 'that computer readable programs embodied in a tangible medium, such as floppy diskettes, are patentable subject matter under 35 U.S.C. § 101 . . . ."); Examination Guidelines *supra* note 1, at 7481 ("When functional descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases."); 35 U.S.C. § 101 (2000).

33. *In re* Beauregard, 53 F.3d at 1584. Under the printed matter doctrine, "a mere arrangement of printed matter though seemingly a 'manufacture' is rejected as not being within the statutory classes." MPEP, *supra* note 3, § 706.03(a).

34. *In re* Beauregard, 53 F.3d at 1584.

35. *Id.*

referred to as *Beauregard* claims, as statutory subject matter (i.e., articles of manufacture).

The following subsections present two possible reasons why the PTO reversed its position with regard to the patentability of computer programs embodied on a computer-readable medium in *Beauregard*. The first possibility is that the PTO interpreted and extended the rationale in the *Lowry* decision regarding the printed matter doctrine. The second possibility is that the overwhelming support from the software industry influenced the PTO's position. In both cases, we suggest that the PTO never considered source code when declaring computer programs embodied in a computer-readable medium to be patentable.

D. Following the Lowry *Rationale*

The reversal of the PTO's position regarding *Beauregard* claims may likely have been motivated by the Federal Circuit's decision in *In re Lowry*.[36] As such, it is important to explore the metes and bounds of the decision in order to understand whether, under *Lowry*, source code (as opposed to object code) on a computer-readable medium infringes *Beauregard* claims.

In *Lowry*, the Federal Circuit upheld patent claims for a memory containing data stored in a data structure.[37] In doing so, the court rejected the Board's assertion that such claims could be analogized and rejected under the printed matter doctrine. Specifically, the Board reasoned that the functional relationship between the printed matter (data stored in the data structure) and the substrate (memory) was not new or non-obvious; in the Board's view, Lowry's invention

---

36. *See* Jeffrey S. Draeger, Comment, *Are* Beauregard *Claims Really Valid?*, 17 J. MARSHALL J. COMPUTER & INFO. L. 347, 361 (1998) ("The *In re Lowry* decision came after the PTO Board's decision on Beauregard's claims but before the claims reached the Federal Circuit. Thus, the decision in *In re Lowry* foreshadowed the reversal of the PTO Board's application of the printed matter rejection in the *In re Beauregard* case, since the Federal Circuit reversed a printed matter rejection in *Lowry*."); Jeffrey R. Kuester et al., *A New Frontier in Patents: Patent Claims to Propagated Signals*, 17 J. MARSHALL J. COMPUTER & INFO. L. 75, 79 (1998) ("It should be noted that the Appellants filed their brief in *In re Beauregard* on April 4, 1994 and the Federal Circuit issued its opinion in *In re Lowry* on August 26, 1994. It appears that the PTO may have decided to allow the application involved in *In re Beauregard* to issue as a patent after receiving the Federal Circuit's decision in *In re Lowry*, since both cases involved the application of the printed matter rejection to claims directed to computer programs stored in a memory device."); GREGORY A. STOBBS, SOFTWARE PATENTS § 9.59 (Supp. 1999) ("Nevertheless, as *Lowry* had extensively addressed the printed matter rejection, the Patent Office moved to remand *Beauregard* for reconsideration, claiming that the rejection of Beauregard's application may have been improper in light of *Lowry*.").

37. *In re* Lowry, 32 F.3d 1579, 1581 (Fed. Cir. 1994).

merely disclosed the storage of information into a computer's memory.[38] In response, the Federal Circuit found that the printed matter doctrine was inapplicable and emphasized that Lowry's claimed data structures defined "functional characteristics of the memory."[39] The court observed that "the claims require specific electronic structural elements which impart a physical organization on the information stored in memory," that "Lowry's data structures impose a physical organization on the data," and that "Lowry's data structures are physical entities that provide increased efficiency in computer operation."[40]

Thus, if the PTO was indeed motivated by the Federal Circuit's rationale in *Lowry* to accept computer-readable medium patent claims as statutory, then whatever "information" is recorded on such a computer-readable medium should satisfy the above *Lowry* requirements. Specifically, the Federal Circuit's rejection of the printed matter doctrine in *Lowry* relied on the fact that the "claims require specific electronic structural elements which impart a physical organization on the information stored in memory."[41] In other words, when *object code* is loaded into the computer, it directs the computer's CPU to physically change the exact sequence of bits stored in the computer's memory, thereby manifesting or "forming" the actual data structures in the memory.[42] Conversely, a memory that is not manipulated by the CPU through the computer instructions in object code would *not* infringe on the *Lowry* claim. Since *source code* cannot be loaded into the computer to manipulate the CPU and manifest such data structures in memory, source code, even when loaded into a computer, could in no way infringe the *Lowry* claim.[43]

The PTO must have believed that, under the *Lowry* court's rationale for rejecting the printed matter doctrine, the Federal Circuit

---

38. *Id.* at 1582.

39. *Id.* at 1583 ("Thus, Lowry's claims define functional characteristics of the memory.").

40. *Id.*

41. *Id.*

42. *See generally* ANDREW S. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION 398 (3d ed. Prentice Hall 1990) ("[T]hree programs—the user's *object program*, the operating system, and the microprogram—can be found in the computer's memory at run time. All traces of the original source program have vanished." (emphasis added)). For a simple example of the implementation of a data structure in memory, which demonstrates the implementation of a stack data structure in memory, see *id.* at 178–86.

43. While loading the source code into the memory of a computer does impart a physical organization on the memory, such physical organization would not realize the functionality as described in the claims.

would have allowed Lowry's claimed data structures *even if* the object code was embodied on a computer-readable medium, rather than *actually* loaded into the computer's memory, thereby creating the data structures in a *different* part of the computer's memory.[44] Thus, although object code is merely an instruction set that can create a data structure in a computer's memory, and not the data structure itself, the PTO did not view this as an important distinction. Whether the underlying substrate was a computer-readable medium or a computer memory was unimportant, as long as what was embodied in the underlying substrate (i.e., object code) would be able to "impart a physical organization on the information stored in memory," *once it was loaded into a computer.*

Under the above rationale, object code embodied on a computer-readable medium would infringe a *Beauregard* claim. Specifically, once object code (i.e., a sequence of ones and zeros that are interpreted as instructions by the computer's CPU) is loaded into a computer, it "impart[s] a physical organization on the information stored in memory" and "define[s] functional characteristics" of the computer's memory by directing the computer's CPU to manipulate data and by referring to specific addresses in the computer's memory.[45] Object code stored on a computer-readable medium can be directly loaded into a particular computer in order to direct the computer's CPU to create "specific electrical structural elements which impart a physical organization on the information stored in memory."[46] Thus, in order for a memory to embody the data structures described in *Lowry*, the code that effects this functionality is *necessarily* object code, not source code.

Unlike object code, source code *cannot* be directly loaded into a computer in order to direct the computer's CPU to create "specific electrical structural elements which impart a physical organization on the information stored in memory."[47] Because source code is human-readable, not computer-readable, it must first be compiled into computer-readable object code before it can direct a computer's CPU

44. It is important to understand that the computer memory in which object code (i.e., instructions) is loaded in order to direct the computer's CPU to manipulate data via data structures is a *different* part of the computer's memory than the memory that the computer's CPU uses to actually create the data structures and manipulate data per those object code instructions.

45. *In re* Lowry, 32 F.3d 1579, 1583–84 (Fed. Cir. 1994).

46. *Id.*

47. *Id.*

to "define functional characteristics of the memory."[48] While object code enables a computer to manipulate specific addresses in the computer's memory (i.e., "impart[s] a physical organization on the data"[49]), source code is simply a description of that capability. As such, one could argue that source code embodied on a computer-readable medium is more like "mere data" that would be rejected as non-statutory under the printed matter doctrine since, in human-readable form, source code is only "useful and intelligible to the human mind,"[50] and has no functional relationship to any structural element or physical organization of a computer, as required by *Lowry*. Under this argument, human-readable source code does not fulfill the requirements under *Lowry*—source code only becomes functional when it is compiled into object code.

E. Support from the Software Industry

Alternatively or additionally, the PTO's change of position in *Beauregard* may have been influenced by the overwhelming support for computer-readable medium claims by the software industry.[51] One of the main reasons that the software industry strongly endorsed the patentability of computer programs embodied on a computer-readable medium was that software patents in the form of process and machine claims were difficult to assert against competitors.[52] Because process and machine claims could only be directly infringed by end users who were typically customers of the software

---

48. *Id.*

49. *Id.*

50. *Id.* at 1583 ("The printed matter cases 'dealt with claims defining as the invention certain novel arrangements of printed lines or characters, useful and intelligible to only the human mind.'" (quoting *In re* Bernhart, 417 F.2d 1395, 1399 (C.C.P.A. 1969))).

51. A total of 10 amicus briefs were filed in *Beauregard*, nine urging reversal of the Board's rejection and one taking no formal position. Shawn McDonald, *Patenting Floppy Disks, or How the Federal Circuit's Acquiescence has Filled the Void Left by Legislative Inaction*, 3 VA. J.L. & TECH. 9, ¶ 89 (Fall 1998), *at* http://vjolt.student.virginia.edu/graphics/vol3/home_art9.html; *see* Robert C. Laurenson, *Computer Software 'Article of Manufacture' Patents*, 12 COMPUTER LAW. 18, 19 (June 1995).

52. *See* Laurenson, *supra* note 51, at 19

> (The alternative means of obtaining protection of computer software under the patent system, *i.e.*, by casting the software in the form of the "process" it performs, or in the form of the "machine" on which it executes, are also not entirely adequate. Claims directed to the process performed by the software (or the machine on which the software executes) are largely directed to the activities of end users. Such claims may thus not be entirely effective for the purpose of enforcing the underlying patent against competitors.).

companies, such companies were reluctant to bring suit.[53] As such, software companies would have to sue competitors for contributory infringement or inducing infringement, both of which required proof of direct infringement by customers as well as knowledge of the patent.[54] Thus, allowing claims for computer programs embodied in computer-readable mediums would enable software companies to sue competitors for direct infringement and eliminate the burden of having to prove indirect infringement.[55]

The above arguments made by the software industry only contemplated the possibility of object code being stored on a computer-readable medium, and not source code. That is, software companies were primarily concerned that competitors would commercialize their patented inventions (i.e., sell executable object code) with impunity. Since the software companies' concerns focused upon a new direct cause of action against competitors, to eliminate the need with process and machine claims to assert an indirect infringement suit, then one could argue that the ability to assert such a new direct cause of action should be construed narrowly, within the jurisprudence and rationale that lies behind the patentability of software as machine and process claims. As discussed earlier, both machine and process claims that involve software must necessarily utilize object code, not source code. It therefore follows that the related *Beauregard* claims should only be infringed when object code, and not source code, is stored on a computer-readable medium. Under this argument, there seems to be no compelling reason to extend patent protection to source code embodied on a computer-readable medium, since source code cannot be loaded into a computer to infringe a machine patent nor can it be executed to infringe a process patent.

One might argue that the transformation of source code to object code through compilation is merely a mechanical process that should make no substantive difference in its treatment.[56] Therefore, the argument continues, if object code in a computer-readable medium infringes a *Beauregard* claim, then so should the original source code. However, this argument misses the point, that an allegedly infringing product must realize the functionality (i.e., the elements) as claimed

53. *See* McDonald, *supra* note 51, ¶ 98.

54. *See* Laurenson, *supra* note 51, at 21 n.22; McDonald, *supra* note 51, ¶ 98. *See also* 35 U.S.C. § 271(b)–(c) (2000).

55. *See* McDonald, *supra* note 51, ¶ 99.

56. Note, however, the complexity of compilers as described *supra* in Section II.

in the patent.[57] While object code can be directly loaded into a computer to realize this functionality, source code cannot.

As an analogy, consider a detailed blueprint for a new, non-obvious, and useful car. Assume that such a blueprint can be simply scanned and submitted to a master computer that directs the robots on the production floor to build the car in accordance with the blueprint. Photocopying (or "making") or selling the blueprint would not infringe any patent rights in the car since the blueprint does not realize the functionality of the claims (nor is it proper statutory subject matter). In contrast, making or selling a car built from those blueprints would likely infringe a machine claim in the patent under the rights granted under 35 U.S.C. § 271.[58] As such, despite being a mere mechanical process from blueprint to car, only the car is proper patentable subject matter, as a machine. Similarly, source code is like a detailed blueprint to the object code (i.e., the car). Since source code is not computer-readable and therefore can not effect the functional or structural requirements under the patent claims, the argument that compiling source code to object code is merely mechanical, like the transformation of the blueprint to the car, does not effectively support the theory of source code patentability.[59]

F. The PTO's Guidelines for Computer-Related Inventions

Nevertheless, the PTO Guidelines issued in 1996 provide examples of computer-readable medium claims comprising "source code segments," indicating that the PTO does consider source code embodied in a computer-readable medium to be statutory subject matter.[60] Specifically, the PTO provides the following example as a proper article of manufacture claim:

---

57. The MPEP recommends that "[A]pplicants should be encouraged to *functionally define the steps* the computer will perform rather than simply reciting source or object code instructions." MPEP, *supra* note 3, § 2106, at 2100-20 (emphasis added).

58. 35 U.S.C. § 271(a) ("Except as otherwise provided in this title, whoever without authority makes, uses, offers to sell or sells any patent invention, within the United States, or imports into the United States any patented invention during the term of the patent therefore, infringes the patent.").

59. One might further argue that the source code/object code distinction is meaningless because the source code already describes the new, non-obvious, and useful invention that a patent owner is trying to protect, and to allow such source code to be copied or distributed with impunity would weaken the strength of patent protection. However, this argument fails to take into account that the very paper upon which a patent specification and claims are written also describes the new, non-obvious, and useful invention that a patent owner is trying to protect.

60. *See* PATENT & TRADEMARK OFFICE, U.S. DEP'T OF COMMERCE, EXAMINATION GUIDELINES FOR COMPUTER RELATED INVENTIONS: CLAIM EXAMPLES—COMPRESSION/ENCRYPTION EXAMPLES (Mar. 28, 1996), *available at*

> A computer program embodied on computer-readable medium for monitoring and controlling an automated manufacturing plant using a telemetered processed data signal comprising:
>
> a. a compression source code segment comprising . . . [recites self-documenting source code]; and
>
> b. an encryption source code segment comprising . . . [recites self-documenting source code].[61]

Seemingly contradictorily, the PTO Manual of Patent Examining Procedure ("MPEP") notes that only "functional descriptive material" and not "nonfunctional descriptive material" is patentable when claimed on computer-readable medium because it "permits the function of the descriptive material to be realized."[62] The MPEP describes "functional descriptive material" as consisting of "data structures or computer programs which *impart functionality when employed as a computer component.*"[63] Arguably, only object code, not source code, meets this description. Source code, without more, when embodied on a computer-readable medium, neither "permits the function it describes [in human-readable form] to be realized" nor can it be "employed as a computer component [which imparts

---

http://www.uspto.gov/web/offices/pac/dapp/oppd/pdf/compenex.pdf [hereinafter CLAIM EXAMPLES]; *see also* Examination Guidelines *supra* note 1, at 7485; MPEP, *supra* note 3, § 2106. MPEP § 2106 includes the Guidelines as well as additional comments regarding computer-related inventions not included in the Guidelines. *Id.*

61. CLAIM EXAMPLES, *supra* note 60, claim 12, at 34. Notice that such a claim would *not* be infringed by a CD-ROM containing object code. Furthermore, MPEP § 2106 states that:

> When a claim or part of a claim is defined in computer program code, *whether in source or object code format*, a person of skill in the art must be able to ascertain the metes and bounds of the invention. In certain circumstances, as where self-documenting programming code is employed, use of programming language in a claim would be permissible because such program source code presents "sufficiently high-level language and descriptive identifiers" to make it universally understood to others in the art without the programmer having to insert comments . . . Applicants should be encouraged to functionally define the steps the computer will perform rather than simply reciting source or object code instructions.

MPEP, *supra* note 3, § 2106, at 2100-19 to -20 (emphasis added). The foregoing advises a patent examiner how to assess whether a claim that is defined in computer program code "particularly pointing out and distinctly claiming the invention" as required under 35 U.S.C. § 112. *See* 35 U.S.C. § 112 (2000). This assessment differs from the assessment of whether the actual source code *itself*, if embodied on a computer-readable medium, is statutory subject matter, under 35 U.S.C. § 101. While using source code to describe the functionality of a claim (e.g., whether a process, machine, or product claim) may prove adequate for § 112 purposes, the foregoing statement says nothing about whether source code itself can infringe a Beauregard claim.

62. MPEP, *supra* note 3, § 2106, at 2100-12.

63. *Id.* at 2100-11 (emphasis added).

functionality]."[64] Only when source code is compiled into object code do such capabilities emerge. For example, only object code can be "employed as a computer component," namely by being loaded into the computer to direct the CPU to manipulate memory.[65] Source code is much more similar to music, literature, art, or photographs that are embodied on a computer-readable medium. Such material is deemed to be nonstatutory "nonfunctional descriptive material," and the computer-readable medium in which it is embodied is "nothing more than a carrier."[66]

Furthermore, the MPEP defines a computer program as "a set of instructions *capable of being executed by a computer*."[67] Again, under this definition, object code constitutes a computer program, but source code does not, since it is *not* capable of being executed by a computer. The MPEP also suggests that a computer program is only statutory when it is encoded on a computer-readable medium because such a medium is "needed to realize the computer program's functionality."[68] However, even when source code *is* encoded on a computer-readable medium, the source code's functionality is still *not* realized until it is compiled into object code and then loaded into the computer.

As such, the PTO's guidance seems to have added more confusion than clarity. The PTO's reason that data structures and computer programs *not* embodied on a computer-readable medium are not statutory is that "they are not capable of causing functional change in the computer"[69] Thus, the PTO believes that once a data

64. *Id.* at 2100-11 to -12.

65. *Id.*

66. *Id.* at 2100-14.

67. *Id.* at 2100-13 (emphasis added).

68. MPEP, *supra* note 3, § 2106, at 2100-13 ("Office personnel should treat a claim for a computer program, without the computer-readable medium needed to realize the computer-programs functionality, as nonstatutory functional descriptive material.").

69. *Id.* In contrast, the MPEP states that "a claimed computer-readable medium encoded with a data structure" does define structural and functional interrelationship between the computer hardware and software components that do permit the data structure's functionality to be realized. *Id* at 2100-13. Such a statement, however, may reflect a misunderstanding by the PTO of the relationship between data structures and computer-readable mediums. That is, data structures, *themselves*, *cannot* be encoded in a computer-readable medium. However, the *object code* that provides instructions to a computer to create such data structures *in the memory* of the computer *can* be encoded on a computer-readable medium. Specifically, the object code is loaded from the computer-readable medium into the computer's memory, which then directs the computer's CPU to create a data structure in *another part of the computer's memory*. The data structure, itself, can only be manifested *within the memory of the computer*. The foregoing explanation reflects the essence of the *Lowry* decision. *See In re* Lowry, 32 F.3d 1579, 1583–84 (Fed. Cir. 1994). In order to be infringed, the *Lowry* memory claim essentially required object

structure or computer program is embodied on a computer-readable medium, it should be *capable of causing functional change in the computer.* However, source code, even when embodied on a computer-readable medium, is no different from source that is *not* embodied on a computer-readable medium. That is, source code is not capable of causing functional change in a computer—only object code is.[70] As such, source code, whether or not on a computer-readable medium, is not computer-readable itself and therefore seems no different than nonstatutory "computer listings *per se*, i.e., the descriptions or expressions of the programs," *even if* it is physically embodied on a computer-readable medium.[71] Indeed, if source code on a computer readable medium is patentable subject matter even though some intermediate processing is necessary before it can be executed (i.e., compiling), then, by extension, so also should be a handwritten source code handwritten on a piece of paper since the only difference is the number of intermediate machine processing steps (i.e., scanning the code on paper, utilizing handwriting recognition software, and *then* compiling). Consequently, to say that source code on computer-readable medium infringes *Beauregard* claims arguably exalts form over substance and could be contrary to the intentions of the Federal Circuit.

IV. IMPLICATIONS FOR THE OPEN SOURCE COMMUNITY

In 1984, Richard Stallman quit his job as a researcher at the MIT Artificial Intelligence Lab to form the GNU project.[72] Frustrated by what he called the "proprietary software social system," that is, the lack of free sharing and community in the computer industry, Stallman launched the GNU project in hopes of reviving the software sharing community that had inspired him during the early 1970s as an AI Lab staff system developer at MIT.[73] The aim of the GNU project was to develop an operating system for which the source code would

code to *already* be loaded into the computer in order to direct the CPU to create the data structure in the memory.

70. Indeed, a general-purpose computer (such as a PC or a Macintosh) typically does not have compilers installed as part of its operating system. As such, providing a user with a CD-ROM having only source code and no object code would be useless, unless the user had also installed a compiler on her system. Thus, one might argue that the definition of a "computer" would *not* include a compiler.

71. MPEP, *supra* note 3, § 2106, at 2100-13.

72. *See* Stallman, *supra* note 5, at 57.

73. *Id.*

be free.[74] In order to ensure that GNU software would be free, Stallman introduced a concept he called "copyleft" in which everyone was given permission to copy, modify, and distribute modified versions of the GNU software, but not to add restrictions of their own.[75] The concept of copyleft was implemented in the form of the GNU General Public License ("GPL") that accompanied GNU software.[76] A year later, the Free Software Foundation was created as a tax-free charity to promote computer users' right to use, study, copy, modify, and redistribute computer programs, including the GNU operating system, as free software.[77]

However, because the GNU project and the Free Software Foundation were decidedly anti-business, a group of leaders in the free software community, including Eric Raymond, Tim O'Reilly, and Bruce Perens, developed the idea for Open Source and the Open Source Definition in 1997 to encourage businesses to adopt the concept of free software.[78] Under the Open Source Definition, programmers were assured of (1) the right to make copies of an open source program and distribute those copies, (2) the right to have access to the software's source code, and (3) the right to make improvements to the program.[79] While the GPL satisfied the requirements of the Open Source Definition, it was more restrictive. For example, the Open Source Definition allowed a user of open source software to produce a derivative work and distribute such work as proprietary.[80] In contrast, the user of GNU software, under the GPL, was required to distribute derivative works under the terms of the GPL (i.e., not proprietary).[81]

Because third parties outside the stream of open source development can hold software patents related to the techniques and

---

74. Under Stallman's definition, a program is free if: (1) you have the freedom to run the program for any purpose, (2) you have the freedom to modify the program to suit your needs, (3) you have the freedom to redistribute copies; and (4) you have the freedom to distribute modified versions of the program. *Id.* at 56.

75. *See id.* at 59.

76. *See* FREE SOFTWARE FOUNDATION, INC., THE GNU GENERAL PUBLIC LICENSE (GPL) (Version 2, June 1991), *at* http://www.opensource.org/licenses/gpl-license.html.

77. For more information, see Free Software Foundation, *at* http://www.fsf.org/fsf/fsf.html (last modified Apr. 8, 2002).

78. *See* Chris DiBona et al., *Introduction* to OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 5, at 3.

79. *See* Bruce Perens, *The Open Source Definition, in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION, *supra* note 5, at 172.

80. *See id.* at 177.

81. *See id.*

functionalities utilized in an open source project, software patents have always been viewed as a threat to free software by the open source community.[82] The essential fear of the open source community is that free software development and distribution can be controlled or prevented by third party patent holders. However, if source code, as discussed earlier, is not patentable, such fears, with regard to source code development and distribution, can be allayed. That is, no third party patent can prevent open source developers from copying, modifying, or distributing source code.

A. Non-Infringing and Infringing Open Source Activities

Those in the open source community typically describe the consequences of impeding on software patent rights broadly, without exploring in detail how certain open source activities might, if at all, infringe on such rights. For example, Russell Pavlicek writes that "[i]f the author unknowingly violated a software patent, the program cannot be *distributed* without permission from the patent holder."[83] Similarly, Richard Stallman laments that "[s]oftware patents monopolize an algorithm, or a feature, or a technique so that nobody [but the patent holder] can use them in developing a program. And this makes software *development* dangerous."[84]

Despite such concerns about infringement, it is clear that if source code embodied in a computer-readable medium cannot infringe a *Beauregard* claim, then the distribution and development of source code, as well as the studying, copying, and modification of source code, without more, cannot infringe any patent. As discussed earlier, the pertinent statutory subject matter that can be patentable in the computer software arts are processes, machines, and articles of manufacture.[85] In order to infringe a patented process, one must necessarily practice the steps in the process. However, a patented process regarding software can only be practiced when the object code, not source code, is executed on a computer, thereby realizing

82. *See, e.g.*, Stallman, *supra* note 5, at 67 ("The worst threat we face comes from software patents, which can put algorithms and features off-limits to free software for up to twenty years.").

83. PAVLICEK, *supra* note 5, at 161 (emphasis added).

84. J.S. Kelly, *An interview with Richard Stallman*, LINUXWORLD.COM, 2000 (emphasis added), *at* http://www.linuxworld.com/linuxworld/lw-2000-03/lw-03-rms.html?4-4 (last visited Apr. 11, 2002).

85. 35 U.S.C. § 101 (2000) ("Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.").

the functionality of the process. Similarly, a patented machine regarding software can only be infringed when the object code, not source code, is loaded into the memory of a computer.[86] Finally, under *Beauregard*, a computer program embodied on a computer-readable medium can be patentable as a product or article of manufacture. However, if, as argued earlier, a "computer program" must be in object code format in order to satisfy the elements of a *Beauregard* claim, then activities regarding source code are also free from the possible infringement of patented products (i.e., *Beauregard* claims).

While activities regarding source code may be free from patent concerns, the same cannot be said for activities regarding object code. That is, under the Patent Act, direct infringement consists of making, using, offering to sell, or selling the invention defined by the claims of a patent, without the authority of the patent owner.[87] Under a software process patent, running the object code on a computer could constitute "using" the invention under the Patent Act. Similarly, under a software machine patent, loading the object code into a computer's memory could constitute "making" the invention under the Patent Act. Indeed, even simply copying the object code onto a CD-ROM, floppy diskette, or hard disk drive or just compiling the source code into object code and saving the object code onto a CD-ROM, floppy diskette, or hard disk drive could constitute "making" the invention under a *Beauregard* claim. Similarly, distributing the object code to third parties could constitute "selling" the invention under a *Beauregard* claim.

In summary, if source code does not infringe *Beauregard* claims, those open source activities that involve only source code may be free from patent infringement concerns. Under this premise, the practice of open source security—that is, the widespread distribution of the source code of security software in an effort to study and quickly identify vulnerabilities in the code—does not implicate software patent rights, since the distribution of source code does not impede

86. *See In re* Alappat, 33 F.3d 1526, 1545 (Fed. Cir. 1994) ("[A] computer operating pursuant to software may represent patentable subject matter, provided, of course, that the claimed subject matter meets all of the other requirements of Title 35.").

87. *See* 35 U.S.C. § 271(a) ("Except as otherwise provided in this title, whoever without authority makes, uses, offers to sell or sells any patented invention, within the United States or imports into the United States any patented invention during the term of the patent therefor, infringes the patent.").

any patent rights.[88] Similarly, the pure development or modification of source code, whose functionality may be claimed in a software patent, does not infringe on such patents.[89] Likewise, general activities regarding source code, such as copying, modifying, and distributing, which lie at the heart of the open source movement, do not infringe on software patents. Nevertheless, the moment that source code is compiled into object code, or object code, rather than source code, is run, copied, distributed, or modified, then software patent rights may be implicated.

B. Liability Under Contributory Infringement or Inducing Infringement

Even if source code embodied on a computer-readable medium does not infringe *Beauregard* claims, those who copy, modify, or distribute source code must be aware of potential liabilities under theories of contributory infringement or induced infringement. The Patent Act defines contributory infringement as selling:

> [A] component of a patented machine, manufacture, combination or composition, or a material or apparatus for use in practicing a patent process, constituting a material part of the invention, knowing the same to be especially made or especially adapted for use in an infringement of such patent, and not a staple article or commodity of commerce suitable for substantial noninfringing use.[90]

Thus, distributing or selling source code could constitute contributory infringement, since source code could be considered "a component of a patented manufacture" in which the patented manufacture (i.e., *Beauregard* claim) is the compiled object code (embodied on a computer-readable medium).[91] However, contributory infringement requires that the alleged infringer have knowledge of the patent, as well as knowledge that that compilation

88. *See, e.g.*, Alex Salkever, *Is Open-Source Security Software Safe?*, BUSINESSWEEK ONLINE, Dec. 11, 2001, *at* http://www.businessweek.com/bwdaily/dnflash/dec2001/nf20011211_3015.htm (last visited Apr. 11, 2002).

89. Nevertheless, such development or modification almost always necessarily involves compiling the source code into object code for testing purposes. Such compilation could potentially infringe a *Beauregard* claim in a software patent. However, one might argue that the damages for such infringement would be minimal, if at all.

90. 35 U.S.C. § 271(c).

91. Similarly, source code could also be considered a component of patented machine or process, once it is compiled into object code and either loaded into a computer or executed.

of the source code into object code would infringe the patent.[92] Thus, if the alleged infringer has no knowledge that the object code resulting from compilation of source code may infringe a patent, she will not be liable for contributory infringement.[93] Furthermore, the alleged infringer may also argue that the study and analysis of the source code, *without its compilation*, is a "substantial noninfringing use" which prevents its distribution from implicating contributory infringement.[94]

Similarly, the Patent Act states that "whoever actively induces infringement of a patent shall be liable as an infringer."[95] Thus, distributing or selling source code may be considered "actively inducing infringement" if such distribution or sale leads to the direct infringement of a software machine, process, or manufacture claim (i.e., by utilizing the resulting object code). However, liability for inducing infringement requires that the alleged inducer have the specific intent to encourage direct infringement and not merely that she had knowledge that the acts may constitute infringement.[96] As such, without knowledge of the patent, a distributor or seller of source code cannot be liable for inducing infringement.

V. CONCLUSION

From a computer programmer's perspective, making a distinction between source code and object code may, initially, seem ridiculous. However, from a *legal* perspective, equating the two, at least for patent purposes, may lead to unintended consequences many computer programmers would find objectionable. In our current legal framework, it is clear that executing object code, or even copying object code onto a hard drive, may potentially infringe a software patent. However, as discussed, it is unclear whether the copying of source code also potentially infringes a patent. In a world where source code *is* statutory subject matter and *does* infringe such patents, any programmer who downloads source code from a Web site simply

92. For a general description of contributory infringement, see ROBERT L. HARMON, PATENTS AND THE FEDERAL CIRCUIT § 6.4, 306–12 (The Bureau of National Affairs, Inc., 4th ed. 2001).

93. *See* Hewlett-Packard Co. v. Bausch & Lomb, Inc., 909 F.2d 1464, 1469 n.4 (Fed. Cir. 1990) ("Although not clear on the face of the statute, subsequent case law held that § 271(c) required not only knowledge that the component was especially made or adapted for a particular use but also knowledge of the patent which proscribed that use.").

94. 35 U.S.C. § 271(c) (2000).

95. *Id.* § 271(b).

96. *See* Manville Sales Corp. v. Paramount Sys., 917 F.2d 544, 553 (Fed. Cir. 1990).

to read it and study its quality may be liable for patent infringement. In such a world, it does not matter that the programmer ultimately chooses *not* to use the source code (i.e., compiling it and executing the object code), because the mere act of downloading the source code (i.e., "making" under the Patent Act) itself infringes the patent. Framed in this context, distinguishing source code from object code seems much less ridiculous.

This Article has presented an argument that certain activities relating only to source code, such as copying, modifying, and distributing, may not infringe any third party software patent rights.[97] Specifically, process and machine patents cannot be infringed until object code is either executed or loaded into the memory of a computer, and therefore they are not implicated by activities relating only to source code. Additionally, under *Beauregard* claims, "computer programs" that are embodied in a computer-readable medium could be narrowly construed to mean only object code, since object code, and not source code, is the only format "capable of being executed by a computer."[98] The implication of this interpretation for the open source community is that activities that involve only source code, and not object code, such as open source security efforts, may be freely practiced without the concern of infringing software patents. Nevertheless, any time object code is implicated in an open source activity, software patents still remain "the monster hiding under every software developer's bed."[99]

97. This Article has not addressed the issues regarding interpreters, which immediately execute high-level languages *without compilation*. *See* WEBOPEDIA, INTERPRETER, *at* http://www.webopedia.com/TERM/i/interpreter.html (last modified Dec. 10, 2001). However, without extensive analysis of these issues, if a high-level language that is intended to be executed through an interpreter rather than compiled into object code is embodied in a computer-readable medium, then under the reasons presented in this Article, such an embodiment would likely infringe a *Beauregard* claim.

98. *See* MPEP, *supra* note 3, § 2106, at 2100-13.

99. *See* ROSENBERG, *supra* note 5, at 240.