The Columbia

SCIENCE AND TECHNOLOGY LAW REVIEW

www.stlr.org

THE DISCLOSURE OF SOURCE CODE IN SOFTWARE PATENTS: SHOULD SOFTWARE PATENTS BE OPEN SOURCE?

Kenneth Canfield[*]

Decisions by the Court of Appeals for the Federal Circuit and the Supreme Court have granted patent protection to software, a discipline once thought ineligible, and the number of software patents has skyrocketed. Software differs from more traditional patentable subject areas because software inventions are implemented through code, or logical instructions, rather than through physical structure. The eligibility of software raises two interesting questions: Do existing disclosure requirements require applicants to disclose their code? If not, should the disclosure of code be required under a new disclosure requirement? This note discusses how the United States Patent and Trademark Office and the Federal Circuit have concluded that the current disclosure requirements of written description, enablement, and best mode generally do not require the disclosure of code. However, it argues that worthy justifications exist for requiring disclosure under the written description and best mode requirements. This note thus proposes a new requirement to disclose code due to the nature of software development, and examines the potential consequences of such a disclosure requirement.

I. INTRODUCTION

The recent technological boom has dramatically altered the patent landscape. Decisions by the Court of Appeals for the Federal Circuit (hereinafter the "Federal Circuit") and the Supreme Court have granted patent protection to software, a discipline once thought ineligible, and the number of software patents has skyrocketed.[1] Software differs from more traditional patentable subject areas because software inventions are implemented through code, or logical

[*] Columbia Law School, J.D., Candidate 2006. Submissions Editor, Columbia Science & Technology Law Review, Vol. VII. The author would like to thank Professor Harold Edgar, the Julius Silver Professor in Law, Science, and Technology, for his comments and suggestions.

[1] *E.g.,* Robert M. Hunt, *You Can Patent That? Are Patents on Computer Programs and Business Methods Good for the New Economy?*, Business Review - Federal Reserve Bank of Phila., at 5, 8. (Q1 2001), *available at* http://www.phil.frb.org/files/br/brq101bh.pdf (discussing the increasing number of software patents).

instructions, rather than through physical structure.[2] Therefore, software borders on the traditionally unpatentable categories of "laws of nature, natural phenomena, and abstract ideas."[3] Patents are viewed as a protection-for-disclosure bargain instituted "[t]o promote the Progress of Science and useful Arts."[4] The eligibility of software raises two interesting questions: Do existing disclosure requirements require applicants to disclose their code? If not, should the disclosure of code be required under a new disclosure requirement?

In order to answer these two questions, Part II of this paper introduces the subject of software patents. Part III discusses how the United States Patent and Trademark Office (hereinafter the "USPTO") and the Federal Circuit have concluded that the three disclosure requirements of the Patentability of Inventions Act[5] (hereinafter the "Patent Act") (i.e., written description, enablement, and best mode) generally do not require the disclosure of code. While this conclusion appears reasonable, worthy arguments exist for requiring disclosure under the written description and best mode requirements. Part IV proposes a new requirement to disclose code due to the nature of software development, and examines the potential consequences of such a requirement to disclose code.

II. Software Patents

A. *An Introduction to Code*

The Second Circuit Court of Appeals (hereinafter the "Second Circuit") has explored the significance of code, dividing "code" into source code (the code written by humans) and object code (the code executed by machines):

> Computer languages have been written to facilitate program writing and reading. A program in such a computer language–BASIC, C, and Java are examples–is said to be written in "source code." Source code has the benefit of being much easier to read (by people) than object code, but as a general matter, it must be translated back to object code before it can be read by a computer. . . . Since computer languages range in complexity, object code can be placed on one end of a spectrum, and different kinds of source code can be arrayed across the spectrum according to the ease with which they are read and understood by humans.[6]

[2] *E.g.,* Robert Plotkin, *Computer Programming and the Automation of Invention: A Case for Software Patent Reform*, 2003 UCLA J. L. & Tech. 7(2), at page 4 (2003) http://www.lawtechjournal.com/articles/2003/07_040127_plotkin.pdf.

[3] *Diamond v. Diehr*, 450 U.S. 175, 185 (1981).

[4] U.S. Const. art. I, § 8, cl. 8. This clause has led many to argue that patents should not be awarded on software because the patents do not promote progress. *See infra* notes 22-23 and accompanying text. This paper does not enter this debate, but discusses the disclosure requirements given that software patents are granted.

[5] 35 U.S.C. § 100 *et seq.*

[6] *Universal City Studios, Inc. v. Corley*, 273 F.3d 429, 439 (2d Cir. 2001).

Source code has a dual role as a means of communication and as a functional language. The Second Circuit acknowledged this in the above statement and by noting that "programmers communicating ideas to one another almost inevitably communicate in code, much as musicians use notes."[7]

The predecessor to the Federal Circuit, the Court of Customs and Patent Appeals (hereinafter the "CCPA") characterized the task of writing code as follows:

> In general, writing a computer program may be a task requiring the most sublime of the inventive faculty or it may require only the droning use of a clerical skill. The difference between the two extremes lies in the creation of mathematical methodology to bridge the gap between the information one starts with (the "input") and the information that is desired (the "output").[8]

Like translating from German to English, the court continued, conversion from a "complete thought" in English and mathematics into computer code "is necessarily a mere clerical function to a skilled programmer."[9]

The experiences of programmers indicate that developing a "complete thought" contemplated by the CCPA prior to writing code may be difficult. Dan Bricklin, co-developer of VisiCalc, the first spreadsheet program for a personal computer, described the software development process as a "constant iterative processes" where "[e]very time you test, you end up changing your design, your constraints [such as the amount of memory required or the speed of executing an operation], or your statement of the problem."[10] "You can't just specify [a product], give it to a coder, and say it will work," Bricklin added, explaining that this would result in "lousy programs."[11] "We try to build things," explains Randall Davis, former associate director of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, "and we really don't know what they are until we start to build them. So, one punch line here is, it isn't the programming that is hard; it is figuring out what we're trying to do that is hard."[12] Davis further explains, "There is almost no way to visualize software. . . . Sure, we have flow charts, we have data-flow diagrams, we have control-flow diagrams, and everyone knows how basically useless those are. Flow charts and documentation you write afterward–because management requires them, not because they are a useful tool."[13] The fact that the computer software industry appears to routinely miss expected release dates[14] provides support for

[7] *Id.* at 448.

[8] *In re Application of Sherwood*, 613 F.2d 809, 816-17 (C.C.P.A. 1980).

[9] *Id.* at 817 n.6.

[10] Computer Science and Telecommunications Board, National Research Council, *Intellectual Property Issues in Software*, at 45 (National Academy Press 1991) (alterations in original).

[11] *Id.*

[12] *Id.*

[13] *Id.*

[14] *See, e.g.,* Joris Evers, *Longhorn Looks at Another Delay*: *Release date for the next version of Windows may be pushed back again*, PC World, July 30, 2004, *available at* http://www.pcworld.com/news/article/0,aid,117174,00.asp (Microsoft planned to release the first beta of Windows Longhorn (Vista) in 2004, but delayed until July of 2005); Paula Rooney, *Windows Vista: It's A Date*, CRN,

Bricklin's and Davis's position and calls into question the contrary expert testimony presented to courts, particularly where the experts did not attempt to write the software. A programmer's "inventive faculty"[15] appears to come into play more often than the CCPA imagined. Bricklin's and Davis's comments will be revisited throughout this paper.

B. *The Eligibility of Software Patents: What is the Invention?*

Section 101 of the Patent Act provides that "[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."[16] Under the USPTO guidelines and Federal Circuit opinions, computer patents fall under the Patent Act through two routes: physicality or utility. A computer-related process claim must either: "(A) result in a physical transformation outside the computer for which a practical application in the technological arts is either disclosed in the specification or would have been known to a skilled artisan . . . or (B) be limited to a practical application within the technological arts."[17] Though the Federal Circuit has historically been more lenient when the underlying process is linked to physical structure and claimed as a machine, route (B) is reflected in its recent decisions, which hold patentable a claim if it "produces a concrete, tangible and useful result."[18]

The Federal Circuit's treatment of software patents makes it clear that it views the invention not merely as source code, but rather as the useful function the software accomplishes.[19] The invention "is not in the details of the program writing, but in the apparatus and method whose patentability is based on the claimed combination of components or steps."[20] In *State Street*, the financial system invention is not the "hub-and-spoke system" code, but the "transformation of data, representing discrete dollar amounts, by a machine through a series of

December 2, 2005, *available at* http://www.crn.com/sections/microsoft/microsoft.jhtml?articleId=174900096 (Microsoft releases first beta lacking key features, cuts some features for final version, and may have to cut more to meet release date.).

[15] *In re Application of Sherwood*, 613 F.2d 809, 816-17 (C.C.P.A. 1980).

[16] 35 U.S.C. § 101 (1952).

[17] *Manual of Patent Examining Procedure* § 2106, United States Patent Office, Ed. 8, Rev. 1 (Feb. 2003) [hereinafter *MPEP*].

[18] *Id.* (citing *AT&T Corp. v. Excel Commc'ns, Inc.*, 172 F.3d 1352, 1358 (Fed. Cir. 1999) (concerned with a claimed process)). The guidelines note that a similar analysis applies to machines. *Id.* (citing *State St. Bank & Trust Co. v. Signature Fin. Group, Inc.*, 149 F.3d 1368, 1373 (Fed. Cir. 1998)).

[19] *But see* Fed. Trade Comm'n, *To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy – A Report by the Federal Trade Commission*, at ch. 3, page 49 (Oct. 2003), *available at* http://www.ftc.gov/os/2003/10/innovationrpt.pdf (Robert Young, Chairman of the Center for Public Domain and of Red Hat argued that "we have to require that the person applying for the software patent files the source code behind that patent, because the source code is the invention.").

[20] *N. Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 941 (Fed. Cir. 1990). *See also* Amir A. Naini, *Convergent Technologies and Divergent Patent Validity Doctrines: Obviousness and Disclosure Analyses in Software and Biotechnology*, 86 J. Pat. & Trademark Off. Soc'y 541, 558 (July 2004).

mathematical calculations into a final share price."[21] The software is an implementation of the invention.

This position aligns well with the structure of the patent system for two reasons. First, the patent system would fail to provide the proper incentive to investors by protecting only implementations represented by already-written source code. Assuming software patents promote innovation,[22] inventors would put in less effort to innovate and would be less likely to disclose discoveries if their competitors could avoid infringement by writing new code to implement the underlying processes. Second, writing source code based on a suitable disclosure of the inputs, outputs, functions, and goals is often believed by courts to be within the capacity of one of ordinary skill in the art. If this is true, the key inventive step is developing the disclosure, rather than writing the code. If the implementation in source code but not the underlying process is protected, then the patent would be awarded on what is obvious to one of ordinary skill in the art, thereby violating 35 U.S.C. § 103(a). Brickin's and Davis's views may fit with the view that the code itself is not the invention. They do not argue that the code itself is the invention; instead, they claim that until the code is written the inventor does not know what he has invented. On the other hand, if one were to consider the code itself the invention, these arguments would be less persuasive.[23]

III. THE DISCLOSURE REQUIREMENTS

Section 112 ¶ 1 of the Patent Act provides three requirements to ensure an inventor holds up his end of the patent bargain: written description, enablement, and best mode.

> The specification shall contain a [1] *written description* of the invention, and of the manner and process of making and using it, in such full, clear, and exact terms as to [2] *enable* any person skilled in the art to which it pertains, or with which it is most nearly connected, to *make and use* the same, and [3] shall set forth the *best mode contemplated* by the inventor of carrying out his invention.[24]

These requirements will be discussed in turn, and the paper will explain why a patentee often, but probably not always, can satisfy all three requirements without disclosing functional code.

[21] *State St.*, 149 F.3d at 1373. Although the invention is classified as a "machine," the court emphasizes that it doesn't matter which category of statutory subject matter the machine is directed to, so it is the conversion and achieving the useful result–the final share price–that really matters, whether claimed as a machine programmed with software implementing the process, or directly as a process. *Id.* at 1375.

[22] Whether this is indeed the case is highly contested. *E.g.*, Hunt, *supra* note 1, at 14 ("[T]here are good reasons to expect that such patenting [of computer programs and business methods implemented via computers] will not provide a whole lot more incentive to innovate than these firms already have."); Fed. Trade Comm'n, *supra* note 19, at ch. 3, page 44 (Many panelists asserted that "software and business methods patents . . . are often questionable and are actually stifling innovation by increasing entry barriers and creating pervasive uncertainty.").

[23] If the invention of a software patent is the code itself, this may bolster the argument that patents are not proper for protecting software, because rewarding a patent on the code itself would not provide strong protection.

[24] 35 U.S.C. § 112 (1952) (emphasis added).

A. *Written Description*

The written description requirement ensures that an applicant "convey[s] with reasonable clarity to those skilled in the art that, as of the filing date sought, he or she was in possession of *the invention*."[25] However, "possession alone" is not necessarily sufficient,[26] as the requirement is satisfied "by the patentee's disclosure of 'such descriptive means as words, structures, figures, diagrams, formulas, etc., that fully set forth the claimed invention."[27] This requires more than enablement, which merely requires that an applicant disclose enough so that one skilled in the art can "make and use" the invention.[28]

Recently, the Federal Circuit has stringently applied the written description requirement to biotechnology patents. In *Regents of the University of California v. Eli Lilly Co.*, the Federal Circuit held that the district court did not err in holding a claim invalid for failing to provide a written description of human cDNA, explaining that the disclosure failed to provide "sequence information indicating which nucleotides constitute human cDNA."[29] The Federal Circuit later applied this standard in *Enzo Biochem, Inc. v. Gen-Probe Inc.*, holding that the deposit of claimed sequences in a public depository satisfied the written description requirement of *Eli Lilly*.[30]

The Patent and Trademark Office (hereinafter the "PTO") guidelines and court decisions indicate a more lenient written description requirement for software patents, despite Judge Radar's concern that *Eli Lilly* might require software patents to "disclose every potential coding variation that performs a claimed function."[31] The PTO guidelines explain that "how the specification accomplishes this [showing the inventor had possession] is not material,"[32] and that functional descriptions of software will suffice.[33] The CCPA explained that if the "bridge-gapping tools [i.e., the mathematical methodology between the inputs and outputs] are disclosed, there would seem to be no cogent reason to require disclosure of the menial tools [i.e., the code] known to all who practice this art."[34] Although the Federal Circuit has not set down guidelines,

[25] *Vas-Cath Inc. v. Mahurkar*, 935 F.2d 1555, 1563-64 (Fed. Cir. 1991).

[26] *Enzo Biochem, Inc. v. Gen-Probe, Inc.*, 323 F.3d 956, 969 (Fed. Cir. 2002).

[27] *Id.* (quoting *Lockwood v. Am. Airlines, Inc.*, 107 F.3d 1565, 1572 (Fed. Cir. 1997)).

[28] *Vas-Cath,* 945 F.2d at 1563.

[29] *Regents of the Univ. of Cal. v. Eli Lilly*, 119 F.3d 1559, 1567 (Fed. Cir. 1997).

[30] *Enzo Biochem, Inc. v. Gen-Probe Inc.,* 323 F.3d 956 (Fed. Cir. 2002), *vacating* 285 F.3d 1013 (Fed. Cir. 2002).

[31] *Univ. of Rochester v. G.D. Searle & Co., Inc.,* 375 F.3d 1303, 1313 (Fed. Cir. 2004) (Rader, J., dissenting). *See also* Dan L. Burk and Mark A. Lemley, *Policy Levers in Patent Law*, 89 Va. L. Rev. 1575, 1652-64 (2003) (suggesting that the written description requirement has been applied as a "super-enablement" requirement for biotechnology inventions in a way that would be "inconceivable in other industries, such as software").

[32] *MPEP*, *supra* note 17, at § 2106.01.

[33] *Id.* at § 2106 (citing *Robotic Vision Sys., Inc. v. View Eng'g, Inc.*, 112 F.3d 1163, 1166 (Fed. Cir. 1997); *Fonar Corp. v. Gen. Elec. Co.*, 107 F.3d 1543, 1549 (Fed. Cir. 1997)). *Robotic Vision Systems* and *Fonar* actually grappled with the best mode requirement, as discussed *infra* Part III.C, but the language concerning the ability to functionally describe software has general applicability.

[34] *In re Application of Sherwood*, 613 F.2d 809, 816-17 (C.C.P.A. 1980).

in *Ziarno v. American National Red Cross* it upheld a jury verdict invalidating claims of a software patent for lack of an adequate written description, but appeared to indicate that meeting the requirement without disclosing source code would be possible.[35] The court found substantial evidence to support the jury's finding that the use of "data packets" was not disclosed, explaining that the "specification does not contain the term 'data packet transferring computer network'" or "refer to data packets that are sent over a computer network via a variety of paths and reassembled . . . at the receiving end." An expert witness opined that "one of ordinary skill in the art would not have discerned the 'data packet transferring computer network'" in the specification.[36] The implication is that the presence in the specification of one of the omitted references would have allowed one of ordinary skill in the art to understand that the specification contemplated the use of data packets, satisfying the requirement. Nowhere does the court emphasize the absence of code.

One objection to the easy-to-satisfy written description requirement for software is that it is inconsistent with the stricter written description requirement for biotechnology inventions.[37] For example, one might argue that providing the DNA sequence is analogous to providing source code. However, Professor Amir Naini contends that "the software case law is less inconsistent with biotechnology cases than it may appear at first glance" despite the stricter disclosure requirement for biotechnology.[38] "In biotechnology cases," he explains, "the court generally confronts situations where development times are measured in years rather than weeks or months. The long development times lead the court to scrutinize biotechnology patents for inadequate disclosure, and to establish seemingly exacting standards."[39] He notes that the Federal Circuit in *Eli Lilly* did not focus on the lower court's observation that the University of California researchers took two years to isolate the human insulin DNA using the method disclosed in their specification.[40] Professor Naini argues that this observation could support a decision to hold the written description inadequate.[41] The Federal Circuit may have chosen to omit this argument because the software cases in which the Federal Circuit performed a similar analysis of research dealt with the enablement requirement.[42]

Despite Naini's point, the amount of time required to produce the desired results may be less important than how the time is spent. In *Eli Lilly*, researchers had to engage in testing to determine which nucleotides constitute human DNA. For software, the case law holds that the

[35] *Ziarno v. Am. Nat'l Red Cross*, 55 Fed. Appx. 553 (Fed. Cir. 2003) (not selected for publication in Federal Reporter and not citable as precedent pursuant to Fed. Cir. R. 47.6).

[36] *Id.* at 556.

[37] *See* Naini, *supra* note 20, at 542-43 (arguing that the Federal Circuit applies a low non-obviousness barrier and stringent disclosure requirements to biotechnology inventions, while applying a strict non-obviousness barrier and low disclosure requirements to software inventions).

[38] *Id.* at 559.

[39] *Id.* at 559-60.

[40] *Regents of the Univ. of Calif. v. Eli Lilly & Co.*, No. MDL DKT. 912, IP-92-0224-C-D/G., 1995 WL 735547, *18 (S.D. Ind. 1995) *aff'd in part, rev'd in part*, 119 F.3d 1559 (Fed. Cir. 1997) (affirming invalidity of U.S. Pat. No. 4,652,525 for failing to meet the written description requirement).

[41] Naini, *supra* note 20, at 552.

[42] *Id.*

task of coding may be simple once the specifications are disclosed. Under the Federal Circuit's and PTO's positions, a skilled programmer may know what is being described in a software patent, even if coding the invention takes a long time. This makes sense if the time spent coding is due to time inherently required to write code rather than time spent experimenting. On the other hand, the researchers in *Eli Lilly* had to experiment to determine the scope of the invention since it was not apparent from the specification itself.

At a minimum, the court's interpretation of the written description requirement makes sense when one accepts the view that coding based on a disclosure is a "mere clerical function." To understand why the disclosure of source code is then unimportant, one must acknowledge the two roles of source code: a means of communication and a way to instruct a computer. Computer programmers often communicate in source code because it is convenient, precise, and well-understood by other programmers. Thus, inventors can choose to communicate by disclosing their source code. However, if a mix between prose, pseudocode (i.e., non-functional language that resembles functional code but is modified so as to make it more readable by programmers), and figures can effectively communicate the scope of the invention to one skilled in the art, there is no need to disclose the code; the functional aspect of the code is not important.

Bricklin's and Davis's comments challenge the position that one does not need to disclose source code in order to adequately describe the claimed invention, as they argue that until the code is written, the bounds of the invention are unknown to the inventor. How, then, could the inventor adequately describe the claimed invention without the code? The functional source code serves as proof that the inventor possessed the invention and adequately described it. This paper now considers two replies to this argument.

First, even if the applicant had to write the code before writing the patent specification, it is not necessary for the applicant to disclose it. Once the code is written, the flow charts and documentation adequately describe the claimed invention. Even if they are not useful for the initial developers, they are a useful tool for describing what has been developed. One counterargument is that without the code one would still not know that the *inventor* possessed the invention, even if the inventor provides flow charts and documentation that ultimately allow for coding. Unlike enablement, discussed in the next section, written description focuses more closely on the inventor's knowledge. It seems, though, that an inventor can provide something non-functional and hence not code, but sufficiently detailed, to prove that she possessed the invention. For example, if other programmers can write the program naturally from the description, one might be willing to conclude that the inventor likely did not luck out, but must have possessed the invention, and perhaps written code, in order to produce such good documentation.

Second, an applicant does not need to actually reduce his invention to practice in order to obtain a patent. Thus, if the invention is the general functionality rather than the source code, disclosing the code is unnecessary.[43] While descriptions and diagrams beforehand often may not allow for the production of commercial-grade source code, they may disclose enough to code the invention in an inefficient and non-ideal manner. Also, it is conceivable that patentable inventions are often general enough that specifications do work, and the difficulties described by Bricklin and Davis would only come into play when adding additional requirements for a specific implementation.

[43] *See Patent Model Expert Still Chasing PTO History*, The Patent and Trademark Society Unofficial Gazette, Jan. 24, 2002, http://www.ptos.org/pages.php?menuid=3&pageid=41 (discussing that a model requirement existed in the past but not today).

In fact, a mix between prose, pseudocode, and figures, written after the fact if so required, may do a better job describing the invention than source code. In one Federal Circuit case, the court opined that "providing the functions of the software was more important than providing the computer code."[44] If examiners had to review their code like traditional disclosures, they would have to learn many programming languages to understand the code presented by various applicants. Heavily commented code may help solve this problem, but as an applicant includes more comments, the comments rather than the code might become the heart of the disclosure. Additionally, full disclosure of source code might bury the novel aspects within code corresponding to the user interface, memory and data management, and other aspects of the software, resulting in a longer patent application. Requiring only code snippets of the important code might address the concern with length, but in some cases it might be challenging to separate the new part of the code from the rest. Disclosing code might make it more difficult for already overworked examiners[45] to determine whether a patent should be granted on an application and cause more work for competitors trying to determine the scope of the invention.

Overall, the Federal Circuit's view that the written description requirement does not mandate the disclosure of source code appears reasonable, though one who takes Bricklin's and Davis's comments seriously and focuses on an inventor's knowledge could argue otherwise.

B. *Enablement*

The goal of the enablement requirement is to ensure that the scope of the patent accords with the inventor's contribution to the art.[46] Enablement hinges on whether a person of ordinary skill in the art, using her knowledge of the art along with the patent disclosure, "could make and use the invention without undue experimentation."[47] Some experimentation is acceptable, as "the patent document is not intended to be a production specification,"[48] but the "amount of required experimentation . . . must be reasonable."[49] For software, "enablement is determined from the viewpoint of a skilled programmer using the knowledge and skill with which such a person is charged,"[50] and a court must ask whether the disclosure enables a skilled programmer "to configure the computer to possess the requisite functionality, and, where applicable,

[44] *Fonar Corp. v. Gen. Elec. Co.*, 107 F.3d 1543, 1546, 1548-49 (Fed. Cir. 1997) (dealing with the best mode requirement, but stating a general view about describing software).

[45] *See, e.g., To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy – A Report by the Fed. Trade Comm'n*, *supra* note 19, at ch. 5, pages 4-5 (discussing a large increase in the number of patent applications leading to very short examination times, increased backlogs, and longer pendency periods).

[46] Martin J. Adelman et al., *Cases and Materials on Patent Law* 439 (West Group 2d ed. 2003).

[47] *N. Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 941 (Fed. Cir. 1990).

[48] *Id.*

[49] *White Consol. Indus., Inc. v. Vega Servo-Control, Inc.* 713 F.2d 788, 791 (Fed. Cir. 1983). *See also MPEP*, *supra* note 17, at § 2106.02.

[50] *N. Telecom*, 908 F.2d at 941. Note that many computer-related inventions will involve more than one field, in which case the "disclosure must satisfy the enablement standard for each aspect of the invention." *MPEP*, *supra* note 17, at § 2106.

interrelate the computer with other elements" without "undue experimentation."[51] The level of detail of disclosure that satisfies this "may vary according to the nature of the invention, the role of the program in carrying it out, and the complexity of the contemplated programming,"[52] but the CCPA has characterized the task of writing computer code to make its disclosure unnecessary.

In fact, far less than program code can be sufficient, as in *Northern Telecom, Inc. v. Datapoint Corp.*, where the absence of detailed flow charts, block diagrams, or source code listings did not stop the Federal Circuit from holding that the disclosure satisfied the enablement requirement.[53] Although the court was satisfied that the program could be written "with ordinary effort," the only indication it provides of what qualifies as "ordinary" is that the program could be written in something less than "one and one half to two person-years of work."[54] In *White Consol. Indus., Inc. v. Vega Servo-Control, Inc.,* the court held that a time requirement of one and one half to two person-years was "clearly unreasonable," and that the disclosure therefore failed enablement because it necessitated undue experimentation.[55] On the other hand, a district court held undue experimentation was not required when a programmer of ordinary skill in the art with no prior familiarity with the invention wrote a program four hours after receiving the application.[56]

The CCPA in *In re Application of Sherwood* as well as the Federal Circuit in *Northern Telecom* have properly analyzed the issue, despite the Bricklin and Davis view that programming from specifications might not be as simple as the courts imply. The reasons discussed in connection with the written description requirement apply here too. In fact, even though Bricklin's and Davis's remarks appear more related to enablement than to written description because they involve using specifications to write code, they may be less relevant here. With written description, functional code can serve as a proxy to show that *an inventor* actually knows the extent of what he invented. Enablement is a lesser requirement; it deals not with what the inventor knows, but with whether he provides enough information for *another* to make and use the invention. If he provides descriptions close enough to code, it should be sufficient. As Davis explains, "it isn't the programming that is hard."[57] The goals of enablement can be satisfied through pure communication, even when the communication must be written after the code; functional code is not necessary.

Additionally, although the time required to write the code is a good proxy for whether one of ordinary skill in the art can write the software without undue experimentation, it is far from perfect. The classical case of undue experimentation is where an applicant is unsure whether a particular implementation will actually work, and where the application serves only as

[51] *MPEP*, *supra* note 17, at § 2106.

[52] *N. Telecom*, 908 F.2d at 941.

[53] *Id.* at 942.

[54] *Id*.

[55] *White Consol. Indus., Inc. v. Vega Servo-Control, Inc.*, 713 F.2d 788, 791 (Fed. Cir. 1983).

[56] *Hirschfeld v. Banner*, 462 F. Supp. 135, 140 (D.D.C. 1978) (J. Malloy, Chief Judge, CCPA, sitting by designation).

[57] *Intellectual Property Issues in Software*, *supra* note 10, at 45.

"invitation" to experiment.[58] In *Enzo Biochem, Inc. v. Calgene, Inc.*, the Federal Circuit held claims invalid for lack of enablement because "the teachings set forth in the specifications provide no more than a 'plan' or 'invitation' for those of skill in the art to experiment practicing antisense in eukaryotic cells; they do not provide sufficient guidance or specificity as to how to execute that plan."[59]

On the other hand, since computer programs are logical processes, after reading the specification of the program one may be confident that the invention works. "Experimentation," that is, testing the idea to see whether it will work, is often not necessary. This does not mean that it might not take a long time to implement. A software invention might take hundreds of thousands or possibly even millions of lines of code to implement,[60] and this will necessarily be time consuming, but it does not change the fact that there might be little doubt the invention will, when implemented, work. It is true that throughout the development process a programmer will probably run the program, have it not work, and fix it. Yet, this is not the same type of experimentation as determining whether a patented biotechnology invention can be practiced for cells other than those disclosed in the application. In this context the programmers are fairly certain that functional code can be written, while in *Enzo* the invention concerned a "highly unpredictable technology."[61]

An objection to the non-disclosure rule is that even if one of ordinary skill in the art can go from communicatory language to the functional language of source code without undue experimentation, doing so is no different than translating to English from a foreign language, which is necessary because applicants must either file in English or include an English translation.[62] That is, one might argue that just as an applicant must make the simple translation from a foreign language to English, an applicant should make the translation from prose to code, even if this translation is also simple for one of ordinary skill in the art. The Supreme Court in *Corley* even compared writing source code from a specification to translating from a foreign language.[63] This objection is faulty for four reasons. First, the language requirement is not derived from enablement, but is mandated by a separate regulation, so it is not appropriate to simply analogize the two translations by the difficulty of translation. Second, submitting an application in a foreign language puts an extra burden on the USPTO, while describing a software invention without submitting code might convey the invention better than actual code, at least under the reasoning of the Federal Circuit. Third, patents are intended to disclose information to those skilled in the art, who for United States patents are mostly English readers, so allowing foreign language submissions would burden many readers. Finally, since any trials will be conducted in English, and each translator will translate somewhat differently from the foreign language, foreign language patents might complicate claim construction.

[58] *Enzo Biochem, Inc. v. Calgene, Inc.*, 188 F.3d 1362, 1374 (Fed. Cir. 1999).

[59] *Id.*

[60] *To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy – A Report by the Fed. Trade Comm'n*, *supra* note 19, at ch. 3, page 44 (The software and internet industries create programs that sometimes have millions of lines of code.).

[61] *Enzo Biochem*, 188 F.3d at 1372.

[62] 37 CFR § 1.52(b)(1)(ii). *See also* General Information Concerning Patents, United States Patent and Trademark Office (Jan. 2005), http://www.uspto.gov/web/offices/pac/doc/general/.

[63] *See supra* text accompanying note 9.

Therefore, the Federal Circuit's view that the enablement requirement does not mandate the disclosure of source code is consistent with this paper's analysis.

C. *Best Mode*

The best mode requirement "restrain[s] inventors from applying for patents while at the same time concealing from the public the preferred embodiments of their inventions."[64] A court will ask two questions: (1) "whether, at the time the inventor filed his patent application, he knew of a mode of practicing his claimed invention that he considered to be better than any other," and (2) if he did contemplate a best mode, whether "the disclosure [is] adequate to enable one skilled in the art to practice the best mode or, in other words, has the inventor 'concealed' his preferred mode from the 'public'?"[65] The first question is "wholly subjective," depending on what the inventor knew at the time of filing, while the second step is "largely an objective inquiry."[66] The inventor is only required to enable what he considered the best mode known at the time of filing, and anything learned after the filing is irrelevant. If he did not contemplate a best mode at the time of filing, the requirement is moot.[67]

Once a court has made a determination that the inventor contemplated a best mode, the court proceeds the way it does with enablement.[68] In the case of software,

> description of . . . a best mode is satisfied by a disclosure of the functions of the software. This is because, normally, writing code for such software is within the skill of the art, not requiring undue experimentation, once its functions have been disclosed. . . . Thus, flow charts or source code listings are not a requirement for adequately disclosing the functions of software.[69]

In *Fonar Corp. v. General Electric Co.*, the Federal Circuit held that Fonar's disclosure of the functions of two software routines forming part of the contemplated best mode, without providing the code, enabled one skilled in the art to practice their method of obtaining MRI imaging data for multiple angles in one scan. The court even said that "providing the functions of the software was more important than providing the computer code."[70] Likewise, in *In re Application of Sherwood*, the CCPA held that the best mode was satisfied by a general

[64] *MPEP*, *supra* note 17, at § 2106.01 (citing *In re Gay*, 309 F.2d 769, 772 (C.C.P.A. 1962).

[65] *Chemcast Corp. v. Arco Indus. Corp.*, 913 F.2d 923, 927-28 (Fed. Cir. 1990). *See also*, *Fonar Corp. v. Gen. Elec. Co.*, 107 F.3d 1543, 1548 (Fed. Cir. 1997) (citing the two-part rule from *Chemcast* in the software context).

[66] *Chemcast*, 913 F.2d at 928.

[67] *See, e.g., Glaxo Inc. v. Novopharm Ltd.*, 52 F.3d 1043, 1052 (Fed. Cir. 1995) ("The trial court here correctly noted that this court has 'found that the absence of a showing of actual knowledge by the inventor was dispositive of the defendant's best mode argument' and held that the law 'does not permit using imputed knowledge' in a best mode defense.").

[68] *See Chemcast*, 913 F.2d at 928.

[69] *Fonar*, 107 F.3d at 1549.

[70] *Id.* at 1546, 1548-49.

description of the mathematical equations and a statement that the best mode involved use of a digital computer.[71] In *Robotic Vision Systems, Inc. v. View Engineering, Inc.* the Federal Circuit went even further, holding that the patentee did not fail to comply with the best mode requirement for failing to use the word "software," where the use of software was "plainly apparent to one skilled in the art" and "implicit in the specification."[72] On the other hand, an invention for improved recording on compact discs explicitly disclosing a single waveshape would fail to satisfy the best mode requirement if the inventor, as alleged, viewed a different waveshape as the best mode.[73] Unlike *Robotic Vision Systems*, where there was a gap that could be filled in by a programmer of ordinary skill in the art, or *In re Application of Sherwood*, where the formulas and a statement about writing a computer program were sufficient, "[i]t would not be apparent to one of skill in the art to disregard the waveshape explicitly disclosed in the patent and use a different waveshape."[74]

While the best mode requirement reduces to enablement if an inventor contemplates a best mode, the fact that enablement does not require source code disclosure does not necessarily mean the same is true for the best mode. Whether something closer to source code than what is required for enablement must be disclosed to satisfy the best mode requirement depends on what the inventor contemplates as the best mode of her invention. Not requiring code makes sense if an inventor has no preference about how the best mode functions are coded. But what if she has a reason for preferring a particular coding of the functions over another? Even though one of ordinary skill in the art can write some code to perform the functions without undue experimentation, she might not be able to write code comporting with the inventor's contemplated best mode based only on a functional description complying with enablement. Therefore, to enable one of ordinary skill in the art to make and use the best mode, an inventor may have to disclose something closer to source code, such as pseudocode. Though disclosing functional source code may not be necessary, the disclosure will be closer to functional source code than what is required for most inventions under enablement.

An inventor might try to get out of this strict best mode requirement by arguing that it requires more detailed disclosure than what is claimed, as she is claiming the routine generally, not the specific coding. However, the best mode requirement extends to non-claimed material: "Indeed, most of the cases in which we have said that the best mode requirement was violated addressed situations where an inventor failed to disclose non-claimed elements that were nevertheless necessary to practice the best mode of carrying out the claimed invention."[75] This view is sound. Even if code or pseudocode relating to the claimed aspect of the software is separable from the rest of the code, the inventor would still be hiding her preferred way to practice her invention by withholding how she prefers relating the novel code to the other code.

[71] *In re Application of Sherwood*, 613 F.2d 809, 817 (C.C.P.A. 1980).

[72] *Robotic Vision Sys., Inc. v. View Eng'g, Inc.*, 112 F.3d 1163, 1166 (Fed. Cir. 1997).

[73] *Optical Disc Corp. v. Del Mar Avionics*, 45 Fed. Appx. 887, 887, 897-98 (Fed. Cir. 2002) (not selected for publication in the Federal Reporter and not citable as precedent pursuant to Fed. Cir. R. 47.6) (holding that the new waveshape was not disclosed, but remanding the issue as to whether the inventors in fact considered the undisclosed waveshape to be their best mode).

[74] *Id.*

[75] *Chemcast Corp. v. Arco Indus. Corp.*, 913 F.2d 923, 928 (Fed. Cir. 1990).

With regard to Bricklin's and Davis's remarks on software development, suppose that in order for an inventor to adequately describe the invention, she had to first write some preliminary source code, though she did not have to disclose the code as part of the written description requirement. Must this code be described as a best mode as the only mode the inventor practiced before applying? The question should hinge on whether the inventor subjectively thought that it was the best coding; perhaps it was an easy coding, but inefficient or otherwise problematic. Thus, Bricklin and Davis's objection might provide a reason for mandating the disclosure of already-written source code as a best mode in some circumstances.

Even if inventors were required to disclose their source code as part of the best mode requirement, it is unclear whether this requirement would lead to many more patents containing source code. A patentee only has to disclose a best mode if she subjectively prefers a particular embodiment at the time of filing. The easiest way to avoid the requirement is to not develop a preference before filing. Since a patentee can file before actually reducing her invention to practice, she will have an additional incentive to file before attempting (or thinking much) about particular code. This may hurt the public in the long run through less clear patents, because an inventor may further develop her invention and discover more applications as she tries to implement the invention, and therefore disclose more if she had filed later. Filing at an earlier stage also provides less time to weed out the worthwhile inventions from the bad ones. However, since the best mode requirement only applies to the inventor, she can be kept out of the loop with respect to searching for a best mode, possibly mitigating some negative effects of the earlier filing of patents.

IV. MANDATING THE DISCLOSURE OF SOURCE CODE

A patentee therefore can generally meet the § 112 disclosure requirements without disclosing source code, with a possible exception if she contemplated a specific coding as a best mode. Part IV.A examines effects on the breadth and validity of patents if courts were to interpret the disclosure requirements to require disclosing source code. Part IV.B explains how requiring the disclosure of source code may conflict with the principle that an inventor does not need to actually reduce an invention to practice prior to filing. Part IV.C argues that nonetheless, there are policy reasons based on the nature of the software industry to require disclosure. Part IV.D examines issues surrounding the implementation of a new requirement to disclose source code.

A. *Requiring Source Code as Part of the Disclosure Requirements*

1. Scope of the Patent

If disclosing source code were required, inventors might worry that disclosing particular source code would narrow the breadth of their claims. While the extent to which this might occur depends on a court's application of two seemingly conflicting canons of claim construction – (1) claims should be construed in light of the claims themselves, the specification, and prosecution history,[76] and (2) limitations should not be imported into the claims from the specification[77] – an appropriate application of the doctrine of equivalents, combined with careful drafting by the inventor, should avoid having software patent claims construed overly narrowly regardless of the approach taken.

Judges emphasizing the former canon, which can lead to narrower constructions, might interpret the claims as covering only the disclosed code and, through the doctrine of equivalents, the equivalents of the disclosed code. A programmer of ordinary skill appreciates that there are numerous ways to achieve the same result. The fact that the programmers would have a set of presumably functional code to start with would make the task of coming up with alternative coding schemes even easier. Therefore, the range of equivalents would be broad. To ensure that the doctrine of equivalents is applied favorably, the patentee should emphasize that the disclosed code is only one of many implementations. Therefore, it would not be advisable to use the source code as a replacement for a functional description of the software, because it would leave patentees with little material to dissuade judges anxious to limit the claims to the source code disclosed that they did not disclaim other approaches.

Judges applying the second canon would probably give broader scope to the literal meaning of the claims, making reliance on a broad application of the doctrine of equivalents less important. They would accept that even though a particular coding was disclosed, the disclosed embodiment should not limit the claims. These judges might provide more weight to an accompanying functional description of the code than the judges focusing on the first canon because they take a more holistic view in determining what the applicant disclosed.

The goals of the doctrine make it clear that a broad application for software patents can make sense. The doctrine of equivalents is often premised on the idea that it would have been difficult or impossible for applicants to have literally encompassed a particular embodiment at the time of filing.[78] While perhaps for software it might be possible for the applicants to have disclosed any particular coding, expecting applicants to cover all possible codings is unreasonable, and leads to a huge waste of applicants', PTO's, courts', and future inventors' resources. Even though this would not be a case where language gaps or after-arising technology make it impossible to cover concepts at the time of filing, the logic behind the doctrine of equivalents applies.

Thus, disclosure of source code should not overly narrow the scope of a well-drafted patent as long as the inventor can convince a judge that the disclosed embodiment is one of many.

[76] *E.g., Philips v. AWH,* 415 F.3d 1303, 1312-13 (Fed. Cir. 2005); *Vitronics v. Conceptronic*, 90 F.3d 1576, 1582 (Fed. Cir. 1996); *Markman v. Westview Instruments*, 52 F.3d 967, 979 (Fed. Cir. 1995) *aff'd* 517 U.S. 370 (1996).

[77] *E.g., Phillips*, 415 F.3d at 1323 ("recogniz[ing] that the distinction between using the specification to interpret the meaning of a claim and importing limitations from the specification into the claim can be a difficult one to apply in practice.").

[78] *See Festo Corp. v. Shoketsu*, 535 U.S. 722, 734 (2002).

2. Validity

If an inventor were required to provide functional source code along with the specification, problems might arise if the code contained a bug. With respect to the written description requirement, does an inventor fail to possess the invention if the code cannot run? Consider also the enablement requirement: if it were necessary to disclose working code to enable one of ordinary skill in the art to make and use the invention and the code contained a bug, would the claims be invalid? As most computer users know, even final releases of commercial software often have bugs; thus it would be unfair to hold inventors to the high standard of requiring bug-free code, especially when we expect them to file patents long before their products are marketable. It would be especially unrealistic to expect bug-free code for more novel or complicated inventions where code is lengthy or particularly complex. Additionally, if the PTO or litigation opponent testing the software uses a different compiler (a program that converts from source code written by programmers to object code read by computers) than the inventor, bugs unnoticed by the inventor may surface and cause the program to fail.

Fortunately, the burden might not be as high as it seems. Bugs that would remain in source code submitted with a patent are not likely to be the type of bug that prevents the program from accomplishing its basic intended function. When consumers purchase a new office suite or operating system, the programs work basically as intended. It is only when many users attempt to use the program in various ways that the bugs surface. Also, if the bugs do not prevent operation of the program each time it is run or for any combination of input, the disclosure could still meet the requirements. It is unlikely that an inventor would submit code that had not previously worked for at least some trials with respect to the claimed functions. Additionally, inventors would be given the opportunity at trial to show that fixing the bugs was within the capacity of a person of ordinary skill in the art without undue experimentation. With respect to the compiler issue, a disclosure requirement can require inventors to specify the compiler they used, and perhaps require them to use a readily available one. Alternatively, when faced with a validity challenge, the inventor can prove the code works for his chosen compiler.

Therefore, while in the majority of cases a patent should survive a validity challenge, the code disclosure requirement would present an extra burden on inventors unique to the functional nature of code in order to ensure the patent is deemed valid.

B. *A Requirement to Disclose Source Code Distinct from § 112 ¶ 1 Would Conflict with an Established Principle of Patent Law*

Since source code is functional, writing the source code is equivalent to actual reduction to practice as long as one views the software invention as the idea underlying the code. For mechanical inventions, this would be analogous to requiring inventors to have built a working model of their invention before filing and submitting the working model.[79] However in the

[79] From 1790 to 1880, inventors were required to submit "working models" along with their applications. *Patent Model Expert Still Chasing PTO History*, The Patent and Trademark Society Unofficial Gazette, *supra* note 43. In *Chaining Open Source Software: The Case Against Software Patents*, http://lpf.ai.mit.edu/Patents/chaining-oss.html (last visited Dec. 14 2005), Jason V. Morgan suggests requiring software patents to include source code and raises the point that such a requirement would resemble the old model requirement.

context of mechanical inventions, an inventor is not required to submit a working model, nor is an inventor required to have built the invention before filing. In *Pfaff v. Wells Elecs. Inc.*, the Supreme Court explained:

> It is true that [actual] reduction to practice ordinarily provides the best evidence that an invention is complete. But just because [actual] reduction to practice is sufficient evidence of completion, it does not follow that proof of [actual] reduction to practice is necessary in every case. Indeed, both the facts of *The Telephone Cases* and the facts of this case demonstrate that one can prove that an invention is complete and ready for patenting before it has actually been reduced to practice.[80]

Instead, an invention can be shown to be "ready for patenting" in two ways: "by proof of [actual] reduction to practice . . . or by proof that . . . the inventor had prepared drawings or other descriptions of the invention that were sufficiently specific to enable a person skilled in the art to practice the invention [also known as 'constructive reduction to practice']."[81]

Allowing inventors to apply for patents after conception but prior to actual reduction to practice serves several policy goals of patent protection: it provides a greater incentive to engage in research and to invent, it leads to earlier and increased disclosure, and it encourages an efficient use of resources.[82] The possibility of earlier patent protection arguably provides a greater incentive to research and invent because an inventor can patent an invention before working out the intricate details, allowing him to commit less time and money to a given idea before seeking protection. Earlier filing leads to earlier disclosure, and if early availability increases the incentive to research and innovate, then it leads to more disclosure. Early and increased disclosure makes it easier for people to build on each other's work. Finally, once an inventor has sufficiently developed an invention, it is most efficient for that inventor to organize the efforts to develop the invention into something practical and marketable, rather than having multiple individuals or companies simultaneously trying to actually reduce the same invention to practice.[83] The disclosure requirements ensure a patent is not granted too early by making sure that the inventor understands what he invented and that the disclosure enables one to make the invention even though the inventor might not have actually reduced his invention to practice yet.

In light of this policy, it would seem unfair to hold inventors to a higher standard when filing for software patents than non-software inventions, unless there is a peculiar property of software subject matter justifying the additional threshold. Section IV.C.1 suggests that the way

[80] *Pfaff v. Wells Elecs. Inc.*, 525 U.S. 55, 66 (1998). When Alexander Graham Bell applied for a patent on some of his telephone technology, the model requirement was waived, which became a big issue when he sued for patent infringement three years after the patent issued when it was discovered that he had still not yet built the telephone disclosed in his first patent. *The Patent Model*, http://atcaonline.com/phone/patent.html (last visited Jan. 29, 2005) (alteration in original).

[81] *Pfaff*, 525 U.S. at 67-68.

[82] This last reason is based on the "prospect theory" proposed by Edmund Kitch. *See* Rebecca S. Eisenberg, *Patents and the Progress of Science: Exclusive Rights and Experimental Use*, 56 U. Chi. L. Rev. 1017, 1036-37 (1989), *reprinted in part in* Adelman, *supra* note 46, at 26, 35-58.

[83] *Id.*

the software industry progresses provides a justification. Section IV.C.2 suggests that disclosure of code is necessary to achieve interoperability between various programs.

C. *Reasons to Require Source Code as Part of a New Disclosure Requirement*

1. Making Progress in the Software Industry

The software industry advances by incorporating code from older software. Francis Fisher, one-time adviser to the Educational Technology Group at Harvard Law School, explained, “Most software is an accretion of pieces of software that have been previously developed, used in ways the original innovator never contemplated.”[84] Dan Bricklin commented, “You use code that worked before . . . [and y]ou write new code. You tie it all together with all sorts of different types of glue.”[85] Thus, he cautioned, “[Y]ou have to be careful about protecting [code] that can be used all over the place,” but he does believe the law must encourage developers to innovate.[86] As patent law is only constitutionally justified as far as it promotes progress, one must take these concerns seriously.[87]

One way to achieve the proper balance is to require patentees to submit source code and force them to give up copyright protection in exchange for stronger patent rights.[88] Inventors would face a choice between lengthier copyright protection, which protects the expression of the code but allows for competitors to copy functionality, and shorter patent protection, which with the proposed modifications would protect function but allows others to copy portions of the code as long as they use the code in an unclaimed fashion. Either alternative would be better for progress than patent protection under the current system.

The proposed disclosure requirement would help promote innovation compared to the current system when programmers elect patent protection. The removal of fear of copyright infringement would allow users to take source code from patented inventions and integrate it into a new invention. Though copyright protection on source code is a relatively weak barrier, the threat of suit is probably a deterrent, especially for smaller developers. Copyright protection also leads to time wasted searching for different ways to code the same functionality.

While programmers will still have to worry about patent liability, these developers already had to worry about the risk of infringing patents; this fear is not an artifact of the proposed system. Having source code available might make it easier to infringe, and make infringement easier to prove, but software developers already borrowed ideas from prior work, likely including patented work, in order to “reassemble[] bits and pieces” of “previously

[84] *Intellectual Property Issues in Software*, *supra* note 10, at 47.

[85] *Id. See also To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy – A Report by the Fed. Trade Comm’n*, *supra* note 19, at ch. 3, page 44 (noting that “innovation occurs cumulatively” in the software and Internet industries).

[86] *Intellectual Property Issues in Software*, *supra* note 10, at 47.

[87] *See supra* note 4 and accompanying text.

[88] Morgan, *supra* note 79. The author suggests requiring patentees to disclose source code and requiring that the code be “licensed on a sufficiently ‘open’ license.” For information on open source software and open source licenses, *see generally* the website of the Open Source Initiative, http://www.opensource.org/.

developed" software "in ways the original innovator[s] never contemplated."[89] If programmers borrow portions of source code and use them in ways the patentees never contemplated, they are safe from infringement, just as they would be had the source code not been disclosed and they took ideas from the disclosure. Additionally, since improvements on patented inventions are socially valuable (and patentable), promoting innovation even when the end result cannot be practiced without the approval of a patentee still benefits society.

The proposed requirement may also drive some prospective patentees to seek copyright protection instead due to its weaker disclosure requirements. One does not have to register for a copyright, although registering does provide benefits. If one registers a copyright on software, one need only deposit the first and last twenty-five pages of the program,[90] so copyright holders can render the requirements useless by filling up the disclosed portions with comments or less important portions of their source code. Further, the copyright office has provided special guidelines for source code containing trade secrets, leading to less disclosure.[91]

Due to the weaker protection afforded by copyrights, the number of programmers who forgo patent protection may be small: "Copyright protection is extended only to expressions of ideas, not to the underlying ideas themselves[, and therefore] . . . it permits duplication of function."[92] Copyrights on software are easily avoided by would-be infringers who want to achieve the same function. Additionally, with the ease of reverse-compiling object code, getting source code that looks different than the original, and recompiling, copyright owners often have a hard time proving infringement when there was actual copying. Nevertheless, if developers do elect copyright protection, their underlying ideas are free for the taking. While another developer must take the time to write code, he can build on the idea without fear of liability presented by patent protection.

Thus, due to the nature of the software development process, requiring disclosure of source code coupled with forfeiture of copyright rights would promote innovation.

[89] Francis Fisher, commenting on the software development process, in *Intellectual Property Issues in Software*, *supra* note 10, at 47.

[90] *Id.*

[91] United States Copyright Office. Copyright Registration for Computer Programs (Circular 61), at 2 (June 2002), *available at* http://www.copyright.gov/circs/circ61.pdf.

[92] *Intellectual Property Issues in Software*, *supra* note 10, at 24. *See also Computer Assocs. Int'l Inc. v. Altai, Inc.*, 982 F.2d 693, 707-10, 714-15 (2d Cir. 1992) (employing a "'successive filtering method' for separating protectable expression from non-protectable material," which includes ideas, factors dictated by the specifications of the program, requirements due to efficiency concerns, and elements from the public domain, and concluding that defendant's computer program which performed similar functions to plaintiff's program did not infringe).

2. Achieving Interoperability[93]

Computer users often find that they want to use software from a variety of sources, or to use the output of one program in another. They desire compatibility, also known as interoperability.[94] Although many companies make their programs compatible with those of their competitors out of self-interest since it helps attract users of other companies' products to their own, the incentive structure differs for companies with dominant market positions.[95] The industry has responded by creating standards, but this is an imperfect solution. Tying patent rights to the disclosure of code may do a better job of promoting innovation while achieving interoperability.

Standards achieve compatibility, but with two drawbacks. First, standards can freeze the development and use of technology, as they are difficult to modify once in place.[96] Standards "tend to be the least-common-denominator kinds of solutions."[97] De jure standards fail to represent the latest breakthroughs because of the need for compromise among many actors. De facto standards fail to result in the best option because a single large firm can dominate, and because sometimes the prevailing firm may prevail because of chance occurrences rather than the best solution.[98] Second, large producers with a wide variety of software and a significant share of the market will be less likely to participate in standard setting, as compared to small companies that create specific types of software that they wish to make compatible with other programs.[99] Additionally, companies that have invested heavily in developing a new technology might be reluctant to make it freely available as a standard for fear of competition.[100]

Mandating disclosure of source code in patent applications could allow for compatibility without the drawbacks and difficulties of standards. Software manufacturers could look to the latest published applications and issued patents to learn about the file format and interfaces of their competitors' software, and write their software to achieve compatibility. As long as the patents do not cover the mere ability to read and write to the file format, this would not lead to patent infringement concerns. Big companies would be unable to avoid "participating" in the

[93] Bradford L. Smith and Susan O. Mann of Microsoft have written that "openness" works well with patent law because of the disclosure requirements and broad protection of patents despite disclosure, and acknowledged that as compared to trade secrets, this can help achieve the goal of interoperability. Although their definition of "openness" can include the disclosure of source code, it does not necessarily do so, and they do not mention the possibility of mandating the disclosure of source code. *Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?*, 71 U. Chi. L. Rev. 241, 255-56 (2004).

[94] *Intellectual Property Issues in Software*, *supra* note 10, at 49-50.

[95] *Id.* at 50, 67.

[96] *Id.* at 67-68.

[97] *Id.* at 72.

[98] *Id.*

[99] *Id.* at 71-72. For example, Microsoft has purported to open up its Office standards in order to be considered a formal open standard by ECMA International, but developers and attorneys believe Microsoft's open license terms simply amount to a promise not to sue in certain circumstances and are insufficient. *See* Steven J. Vaughan-Nichols, *Microsoft Drops the Office Open Standard Bell*, eWeek, November 29, 2005, http://www.eweek.com/article2/0%2C1895%2C1894039%2C00.asp.

[100] *Intellectual Property Issues in Software*, *supra* note 10, at 71.

push for compatibility unless they also decided to abandon the pursuit of software patents. Unfortunately, this would not allow for the same level of compatibility where the interface itself or methods for converting to or from the interface are patented.[101] But even where the interface or associated methods are patented, others can still work toward improvements of the patented interface or methods, so disclosure is not without any value.

Even if this proposal achieved the goal of increased disclosure there would still be a problem in that large companies file numerous patents, and smaller developers might be unable to keep up. As a result, the smaller developers might have to wait and see what the larger companies implement, and then update their software to achieve compatibility. While in the long run the small developers would hopefully be able to achieve compatibility if desired, the large companies might have a period where they have no competitors with compatible software.

A second concern is that patentees have no obligation to commercially implement the embodiments in their patent disclosures. However, there is reason to believe that the source code of the commercial embodiments will closely resemble the disclosed source code. Disclosure of source code amounts to an actual reduction to practice, which is not currently required. Even though writing source code is generally a trivial activity for a skilled programmer and does not require undue experimentation, it still can be a time-consuming process. It can force a company to spend a large amount of resources to write two complete and significantly different sets of code.

If a patentee were to employ this "two sets of code" tactic, he would risk running afoul of the best mode requirement if he disclosed code subjectively viewed as inferior to other undisclosed code. However, since the best mode requirement only concerns itself with the inventor's knowledge at the time of filing, companies can avoid this requirement by writing a set of scrap code for filing and determining a better implementation after filing. Also, the company could keep the inventor out of the loop and have her submit scrap code, while a separate programming team, unbeknownst to the inventor, works on good code. However, if this practice became known to the inventor, then the inventor's knowledge that others in the company are working on a best mode might still invoke the best mode requirement, even if the inventor does not know any of the details.

Therefore, increased compatibility is another reason to mandate the disclosure of source code.

D. *Remaining Issues Surrounding the New Requirement*

Having provided a basis for a source code disclosure requirement, this paper now considers two issues raised by the requirement. Part IV.D.1 considers the possibility that the

[101] *E.g.,* Ingrid Marson, *Microsoft slammed over XML patent, ZDNet UK*, May 26, 2005, http://news.zdnet.co.uk/software/applications/0,39020384,39200357,00.htm (displaying concern that U.S. Pat. No. 6,898,604 entitled "XML Serialization and Deserialization" to Microsoft would cover "every possible way of converting between a programming object and an XML file," even though the XML standard is open). Microsoft has granted a license to use its XML schemas for the purpose of achieving interoperability with its file formats in certain circumstances. *Office 2003 XML Reference Schema Patent License*, Microsoft, January 27, 2005, http://www.microsoft.com/mscorp/ip/format/xmlpatentlicense.asp. Microsoft has made the technical specifications, including code samples, available to the public. *XML in Office Development*, Microsoft, http://msdn.microsoft.com/office/understanding/xmloffice/default.aspx (last visited December 14, 2005).

increased burden on software inventors would lead to an increased use of trade secret protection instead of patent protection, which would lead to less disclosure. Part IV.D.2 grapples with the implementation of this requirement.

1. Increasing the Attractiveness of Trade Secret Protection

Like patent protection, trade secret protection extends to the functional processes behind the implementation.[102] Trade secrets have the benefit of lasting as long as the information remains secret. However, trade secrets have a number of drawbacks. First, trade secrets are only good as long as one keeps them secret, and for commercially distributed software this is a daunting task. Maintaining trade secrets has become more difficult as outsourcing work overseas has become more common.[103] When trade secrets are maintained through licensing contracts distributed along with the software it is often infeasible to enforce the numerous contracts. These contracts are routinely ignored.[104] Moreover, if a user ends up with the software without agreeing to such a contract, and is not aware or should not have been aware that the software is protected by trade secret, then the user is not bound. Even if an inventor does successfully maintain the trade secret protection, another company can still independently discover the protected invention and patent it.[105] Finally, efforts to maintain trade secrets lead to inefficient development practices, such as ensuring that no one person sees the entire code of a program.[106] Although having different people work on different portions of the code might be beneficial, programmers may need to see other sections of the code in order to fit the pieces together.

Consequently, patent protection is generally more attractive to software developers than trade secret protection, and there appears to be a movement toward protecting software with

[102] For a general discussion on protecting software with trade secrets, *see Intellectual Property Issues in Software*, *supra* note 10, at 29-31.

[103] *See* Patrick Thibodeau, *Trade Secret Theft: Secured Facilities and Rigorous Employee Screening Can Cut the Risk*, Computer World, Nov. 15, 2004, http://www.computerworld.com/managementtopics/outsourcing/story/0,10801,97434,00.html (last visited Feb. 28, 2005) (Outsourcing firms often create walled compounds resembling government intelligence agencies, and take precautions to minimize turnover among employees.).

[104] *See id.* (Twenty-two percent of software installed on computers in the United States, is pirated, the lowest rate in a recent study conducted by IDC and the Business Software Alliance. The problem is worse elsewhere: in China ninety-two percent is pirated, and in India, seventy-three percent.). Although these figures are more relevant to copyright protection, the pirating also generally involved knowing violations of license agreements. *See also* Ian Fried, *Microsoft Wants To Censor Some Open-Source Postings*, C-Net News.com, May 11, 2000, http://news.com.com/Microsoft+wants+to+censor+some+open-source+postings/2100-1001_3-240422.html (Microsoft urged the Slashdot website to remove posts containing specifications for Windows 2000 and Kerberos Web security technology because Microsoft had only made that information available to people agreeing that the material is “confidential” and a “trade secret.”).

[105] *See, e.g., W. L. Gore & Assocs. v. Garlock, Inc.*, 721 F.2d 1540, 1550 (Fed. Cir. 1983) (finding claims not invalid due to secret use by another more than one year before filing).

[106] Thibodeau, *supra* note 103.

patents.[107] However, some software companies still continue to make strong use of trade secret protection.[108] With the advent of a new disclosure requirement, there may be a resurgence in the use of trade secrets as opposed to patents because many companies are adverse to disclosing source code. Companies with a large market share might rely more heavily on trade secret protection because they believe disclosure might make it easier for competitors to introduce competing but non-infringing products. On the other hand, the scope of a patent might be broader. Depending on how the secret is documented, it might be easier to prove in court what is in a patent than what is kept secret. Companies might be hesitant to risk providing yet undisclosed information about their secret in court in order to prove the scope of the trade secret, and instead will rush to reach a compromise. Patents are also attractive in that they can generate large amounts of revenue through licensing fees and exclusive rights. A company with a trade secret rather than a patent might be more hesitant to license, because it does not want to disclose its secret. In deciding which form of protection to pursue, each individual inventor will have to weigh the risks and benefits of patent law as opposed to the risks and benefits of trade secret law.

While requiring the disclosure of source code might lead to less disclosure, it is not immediately clear whether this will help or hurt innovation in the long run. Shubha Ghosh argued that the equilibrium reached through patent law where two parties disclose their inventions is superior to the equilibrium reached through trade secret law where both keep their inventions secret.[109] Other scholars disagree and argue that more patents are not necessarily better.[110] Hunt, focusing on the total amount of research as opposed to the disclosure versus secret equilibria, argues that "[i]n high technology industries, where innovation is already rapid, . . . relaxing patentability criteria is more likely to reduce research activity" than to spur more research.[111] Under Hunt's view, any decrease in software patents resulting from the proposed requirement may benefit society.

2. Implementing the Requirement to Disclose Source Code

This section examines two questions. First, what should the new disclosure requirement require? Second, how should the sufficiency of the submission be evaluated?

With respect to the first question, Congress will need to amend the Patent Act to add this new requirement. In order to minimize the burden on patentees while ensuring beneficial disclosure to the public, it may be sufficient to only require enough code to implement each claim in accordance with a single embodiment, rather than requiring the patentee to implement

[107] Smith & Mann, *supra* note 93, at 242. The authors suggest that a third era of software intellectual property protection may be emerging: in the first era software was protected primarily by trade secrets, in the second primarily by copyrights, and in the third it may be protected primarily by patents.

[108] The efforts undertaken to maintain trade secrets, discussed *supra* note 103, and accompanying text, may provide some evidence of this fact. Microsoft's actions regarding web postings, discussed *supra* note 104 provides evidence that a major force in the industry has trade secrets.

[109] Shuba Ghosh, *Patents and the Regulatory State: Rethinking the Patent Bargain Metaphor after Eldred*, 19 Berkley Tech. L.J. 1315, 1337 (2004).

[110] Hunt, *supra* note 1, at 11.

[111] *Id.* at 11-12.

all disclosed embodiments. The patentee should be allowed to submit the code in his programming language of choice as long as compilers for that language are readily available. Due to the length of the source code, it would be efficient to allow submissions electronically or on a recordable medium.[112]

With respect to the second question, the source code submission must be policed to assure it comports with the disclosure requirement. One possibility is to have the examiner police the requirement, and then allow defendants to challenge its sufficiency during litigation as part of an invalidity claim. Claims can be rejected by the examiner or declared invalid at trial if the submitted code fails to implement the limitations of the claims. Since examiners are already overworked and the source code listing may be extremely lengthy, it would be problematic to require an extremely rigorous review of the code at the PTO level. As Professor Mark Lemley has explained, "[T]he PTO doesn't do a very detailed job of examining patents, but we probably don't want it to," because "it is much cheaper for society to make detailed validity determinations in those few cases [in which patents are challenged] than to invest additional resources examining patents that will never be heard from again."[113] Thus, it is important to consider the burden of source code review on the PTO.

One possibility is to require the examiner to run the code to determine whether the requirement is satisfied, but the examiner should not be required to examine the source code listing. Instead, the applicant can be required to provide the source code along with an executable version limited to specific examples, such as particular data sets discussed in the specification. The examiner can verify the program works on these data sets. Further examination of the source code would be left to a defendant challenging the submission at trial. While this may make already expensive trials even more burdensome, the incremental effect would likely be less than the increase in work if the analysis were performed at the PTO. This way the analysis would be left to the relatively few cases where a dispute arises. Additionally, each side in a patent trial generally already has an expert witness who would testify on the operation of the code submission.

V. CONCLUSION

Due to the explosion in the number of computer software patents and widespread claims that software patents are harmful to the industry, it is important to examine the practice of issuing these patents. This paper has examined how the written description, enablement, and best mode disclosure requirements have been applied to software patents. It asserts that in light of the goals of the three requirements, the USPTO and Federal Circuit's interpretation that the disclosure of source code is not required is generally reasonable, although in some circumstances there exist arguments in favor of requiring disclosure under the written description and best mode requirements. However, due to the nature of the software industry, there are two strong reasons that source code should be disclosed: (1) the software industry innovates incrementally;

[112] The PTO already provides for submission of computer code by compact disc in certain circumstances. 37 C.F.R. § 1.96; *MPEP*, *supra* note 17, at § 608.05.

[113] *To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy – A Report by the Fed. Trade Comm'n*, *supra* note 19, at ch. 5, page 1 (quoting Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 Nw. U. L. Rev. 1495, 1497 (2001)).

and (2) the disclosure of source code promotes interoperability. For these reasons, Congress ought to mandate the disclosure of software code.