PROTECTING COMPUTER SOFTWARE—ANALYSIS AND PROPOSED ALTERNATIVE

Matt Flinders[*]

Cite as: 7 J. HIGH TECH. L. 71

TABLE OF CONTENTS

[*] Suffolk University Law School, May 2006, J.D.; Boston University, May 2000, M.S., Computer Science. The author would like to thank Scott Akehurst-Moore and the 2006-07 Journal of High Technology Law staff for their developmental and editorial support, and Jennifer Luczkow Markowski for inspiration in the study of intellectual property law.

Copyright © 2007 Journal of High Technology Law and Matthew Flinders. All Rights Reserved. ISSN 1536-7983

I. Introduction

As technology eases the ability to copy others' ideas and expressions, present intellectual property laws have struggled to fulfill their intended purposes to protect and promote art and innovation.[1] Changes to the system are imperative. Computer software is a relatively recent and very unique intellectual property that serves both as an expression of an idea and the idea itself. The founders and early developers of intellectual property law could not have foreseen the advent of this unique form of expression and the subsequent problems protecting it.[2] Our legislators and courts have instead attempted to squeeze software within existing copyright and patent laws in doctrinally conflicting fashions instead of carving out new and much needed doctrines for protecting this unique form of intellectual property. This Note looks at the present ways for protecting software from copying, both here and abroad, and the feasibility of such protection. This paper also proposes an alternative approach to protecting software with the intent that the changes be practical, efficient, and consistent with traditional intellectual property policy considerations.

II. Overview of What Constitutes Software

Software, in its most basic form, is a series of instructions copied into temporary or permanent memory on a computer which may subsequently execute those instructions.[3] Software is produced in a variety of human-readable and other very unreadable "languages."[4]

1. *See, e.g.*, NTP, Inc. v. Research in Motion, Ltd., 418 F.3d 1282, 1317-21 (Fed. Cir. 2005) (holding that wireless email system which performed a step of a method in Canada did not infringe a method claim of a U.S. patent); Metro-Goldwyn-Mayer Studios, Inc. v. Grokster Ltd., 380 F.3d 1154 (9th Cir. 2004) (holding that distributors of peer-to-peer software used for massive copying of copyright music and movies were not subject to liability), *rev'd* 543 U.S. 913 (2005) (holding that liability exists for inducing copyright infringement).

2. *See infra* notes 42-66, 80-88 and accompanying text discussing, respectively, the early developments of copyright law and patent law.

3. ENCYCLOPEDIA OF COMPUTER SCIENCE 1599-1600 (Anthony Ralston et al. eds. 2000) (discussing generally how software consists of instructions loaded into computer memory to make them perform their intended tasks).

4. *Id.* at 1043-1045. Machine language is the lowest level of representation for instructions and data is directly readable by a computer's central processing unit. *Id.* These low level instructions performed by the central processing unit are

A processor or controller in a computer comes with a very basic set of available instructions ("machine language") upon which it operates.[5] Whatever the original form, software must ultimately be reduced to those instructions which the computer "understands" and, when executed, becomes the basis for a "live" process on a computer.[6] Software has been generally categorized from low-level to high-level, indicating the relationship and degree of separation between higher level instructions and the most basic machine level codes.[7] As machines on which software is used become more complicated, layers of higher level "instruction sets" that are combinations of lower level instructions are used to simplify programming and improve the "readability" of software for humans.[8] High-level, human-readable code often includes comments by programmers and may even be accompanied by high-level flow charts and other diagrams.[9]

Examples of programs written in lower-level code include "microcode" or "firmware." Low-level code is typically very simple and used on devices like digital watches or other simple machines.[10] Such programs may be seen as an integral part of the hardware, merely replacing the hardwired components of their predecessors with more dynamic and easily changeable "circuitry."[11] A typical personal computer (PC) generally consists of several layers of programming in which the "operating system" (e.g., Microsoft Windows, Macintosh OS) interacts with the computer at its most basic interface and acts as a bridge to higher-level "applications"

generally limited to simple arithmetic computations and shifting instructions that move data between various locations in computer memory. *Id.*

5. *Id.*

6. *See, e.g.*, ANDREW S. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION 2-4 (4th ed. 1999).

7. *Id.*

8. ENCYCLOPEDIA OF COMPUTER SCIENCE, *supra* note 3 at 1441-43. Software code written in higher-level languages, or procedure-oriented languages, are generally machine-independent and human-readable until translated by a compiler into machine language. *Id.* Examples of such higher-level languages include BASIC, C++, and JAVA. *Id.*

9. ROGER S. PRESSMEN, SOFTWARE ENGINEERING: A PRACTIONER'S APPROACH 373-392 (4th ed. 1997) (illustrating that documentation and communication can be an important aspect to computer software development).

10. *Id.* at 4-5.

11. *See* Pamela Samuelson**,** *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663, 676 (1984) (describing, in general, the relationship between computer hardware and computer code).

(e.g., Microsoft Word). Such higher level programs are typically written in more human readable languages such as C++ or Java. Even higher levels of programming may be written by the end-user for automating basic tasks performed within an application (e.g., macros).[12]

III. Brief Overview of Existing Means of Protecting Software

Protection from software copying is presently available in varying degrees under patent, trade secret, and copyright laws. The unique characteristics of software, however, create troubling conflicts under each doctrine. While trade secret law appears to be well settled and applicable to software, the potential of decompilation and reverse engineering limits the effectiveness of its protection.[13] Software's inherent abstraction challenges patentability while its concurrent embodiment as a form of expression and utility challenge copyrightability.

A. Patent Law

Patent law generally provides protection for a "process" or "machine," assuming it meets the arduous standards of patentability, including novelty and utility.[14] The major problem with applying patent law to software is that software often can be reduced to mathematical calculations or manipulations of data, bringing it

12. *See generally id.* at 676-689 (providing a general overview of the layers of programming abstraction).

13. Decompilation is the process by which lower-level code (e.g., machine code) is transformed into a higher-level form from which the underlying ideas/functionality of the code can potentially be discovered, copied, and adapted. *See* The Decompilation Wiki, http://www.program-transformation.org/Transform/DeCompilation (last visited December 30, 2006) (providing additional background information regarding the process of decompilation).

14. 35 U.S.C. §§ 101-103, 112 (2005). Section 101 defines the scope of subject matter that is patentable and requires that inventions be more than abstract ideas that are also "useful". *See infra* notes 15-16. Section 102 requires that the invention is not already "known" by the public. Section 103 requires that the idea is not obvious in light of what is already known publicly. Nonobviousness may be the most subjective of all the standards, where a combination of existing ideas can render an invention unpatentable if that combination teaches each of the elements of the invention and there are identifiable motivations or teachings in the prior art for combining those existing ideas to obtain the invention. *See* Graham v. John Deere Co., 383 U.S. 1, 17-19 (1966) (establishing the "Graham factors", which are the principal factual inquiries for a finding of obviousness); *Teleflex, Inc. v. KSR Int'l Co.*, 119 Fed. Appx. 282 (Fed Cir.), *cert. granted*, 126 S.Ct. 2965 (U.S. 2005).

dangerously close to "abstraction" and what the Supreme Court previously precluded from patentability under 35 U.S.C. § 101.[15] With the advent of *State St. Bank & Trust Co. v. Signature Financial Group, Inc.* ("*State Street*"), however, software directed toward a sufficiently "useful, concrete, tangible" result may be patentable.[16]

As will be discussed further, patent examiners, would-be patentees, and the courts are substantially challenged in determining when and where the line is crossed between abstraction and concreteness. Furthermore, the relatively hidden mechanisms behind commercially-distributed software make it difficult to establish or discount novelty or non-obviousness, rigorous standards under existing patent law.

B. Trade Secret Law

Trade secret law may be the only body of intellectual property law that does not pose substantial new hurdles with respect to software. Indeed, trade secret law is one of software's most useful forms of protection. Trade secret law generally provides protection for almost any form of knowledge so long as it is not "general knowledge."[17] Trade secret law is particularly useful in aiding the protection of high level source code (e.g., Java, C++, and BASIC) which has the benefit of almost built-in secrecy such that it can remain substantially independent of the machine code into which it is ultimately compiled

15. *See* Funk Bros. Seed Co. v. Kalo Inoculant Co., 333 U.S. 127, 130 (1948) (holding that manifestations of nature, including laws that describe them, are not patentable). In *Funk Bros.*, the Court invalidated a patent for a crop-treating combination of separate types of bacteria that turned out to be compatible in a mixture. *Id.* While the Court acknowledged the discovery of their compatibility was previously unknown, this compatibility was something that already occurred in nature and was not something that was created by man. *Id.* Thus, because the patent claims constituted a manifestation of nature and fell outside the scope of patentable subject matter, the Court invalidated the patent. *Id.*

16. 149 F.3d 1368, 1373 (Fed. Cir. 1998) (holding that an otherwise unpatentable abstract idea such as a mathematical algorithm or computation can be patentable if applied in a "useful," "practical" way). In *State Street*, a patent covering a computer program providing investment analysis and administration was challenged on the grounds that the subject matter consisted of abstract mathematical computations and would therefore be unpatentable subject matter. *Id.* at 1370. However, the court held that since the output or outcome of the program (i.e., specific investment calculations) was "useful" and "practical," the subject matter was sufficiently concrete to fall within patentable subject matter. *Id.* at 1375.

17. *See* Metallurgical Indus. v. Fourtek, Inc., 790 F.2d 1195, 1199 (5th Cir. 1986) (stating that trade secrets must first be secrets).

and distributed.[18] Like a secret composition or formula such as the recipe for Coca Cola, the underlying high-level programming behind a software product may be extremely difficult and costly to extract, especially without the aid of a highly skilled expert.[19]

Even though trade secret law provides fairly broad protection for software, the utilitarian nature of software and the ease of copying it have created unique problems bridging trade secret, patent, and copyright laws. Although decompilation[20] (or reverse engineering) of software may be considered a copyright violation in some circumstances, the "fair use" exception may absolve decompilators intending to understand the independent practice of underlying functions.[21] According to the court in *Sega Enters. Ltd. v. Accolade, Inc.* ("*Sega*"), the extracted "functionality" must be "the only and essential means of accomplishing a given task."[22] When, however, people use reverse engineering as a means of indiscriminate copying and resale existing works, the "fair use" defense is not available.[23] Rather, "indirect" or "intermediate use" that led to the creation of distinct independent works allowed the defendant to escape liability.[24]

18. *See generally* Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 660-61 (4th Cir. 1993) (holding that source code of a computer program distributed in object code remained a trade secret), *cited in* ROGER M. MILGRIM, 1 MILGRIM ON TRADE SECRETS § 1.01 n. 3 (2006).

19. *See generally* 1 MILGRIM, *supra* note 18, § 1.09; David A. Einhorn, *Copyright & Patent Protection for Computer Software: Are They Mutually Exclusive?*, *in* 2 MILGRIM, *supra* note 18, app. 9B.

20. *See supra* note 13.

21. *See* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1520-1523 (9th Cir. 1992) (finding that decompiling a video game program to determine its functional interface was "fair use"). For the purpose of developing video game programs compatible with plaintiff Sega's game system, defendant Accolade reverse engineered Sega's copyrighted game cartridge programming. *Id.* at 1514-15. The reverse engineering entailed copying plaintiff's programming, including the process of decompilation which translated the original machine code into human-readable format. *See id. See also supra* note 4.

22. *Sega Enters.*, 977 F.2d at 1524. In weighing the factors for a "fair use" defense, the court ruled that because the copying was the only feasible way of gaining understanding of the program's functionality, and not merely for the purpose of redistributing the original software, the purpose supported a "fair use" defense. *Id.* at 1522-23.

23. *See id.* at 1522 (stating that if the purpose of copying is directly tied with "commercial use" there is a strong inference against establishing "fair use").

24. *See id.* at 1522 (finding that, although copying was related to a "commercial use," since the "character and purpose" of the copying was to determine functionality and only indirectly related to "commercial use," a defense of "fair use" is supported).

Thus, the challenge to the court in *Sega* was to protect authors of software from indiscriminate copying while attempting to avoid treading into areas traditionally exclusive to trade secret and patent law. The combined functional and expressive nature of software even poses problems in trade secret law that typically did not exist in other forms of technology. Unlike software, traditional reverse engineering such as that directed toward secret formulas, machines, or processes generally does not require acts of potential copyright infringement.[25]

C. Copyright Law

The act of copying software is not meaningfully distinguishable from copying digital music or video content.[26] Unlike the content of music or video recordings, however, there are significant questions about how software fits within copyrightable subject matter.[27] Although 17 U.S.C. § 102(b) states that "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation,"[28] "[t]he legislative history indicates that section 102(b) was intended 'to make clear that the expression adopted by the programmer is the copyrightable element in a computer program...'"[29] Furthermore, "[b]ecause of the hybrid nature of computer programs, there is no settled standard for identifying what is protected expression and what is unprotected idea..."[30]

25. *See* 1 MILGRIM, *supra* note 18, § 1.05 ("As a practical matter, it may be impossible to reverse engineer a computer program without decompiling it to create an equivalent source code version--activities which implicate the copyright owner's exclusive rights to, *inter alia,* copy and create derivative works.")

26. *See generally* Metro-Goldwyn-Mayer Studios, Inc. v. Grokster Ltd., 543 U.S. 913 (2005); In re Aimster Copyright Litigation, 334 F.3d 643 (7th Cir. 2003).

27. *See generally* Einhorn, *supra* note 19.

28. 17 U.S.C. § 102(b) (2003).

29. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1252-53 (3d. Cir. 1984) (quoting H.R. Rep. No. 1476 in reference to the legislative intent behind 35 U.S.C. § 102(b)). *See* note 28 and accompanying text. In attempting to clarify the meaning of 35 U.S.C. § 102(b), the court in *Apple Computer, Inc*. reaffirmed that the idea/expression dichotomy bars protection of ideas per se (including any "system" or "process") but protection will be extended to the expression of those ideas, including those aspects of software code that constitute "expression". 714 F.2d at 1252-53. The court also qualified the idea/expression dichotomy with the related "merger" doctrine, which makes the protection of expression contingent on the condition that there are a "number of ways" for expressing the idea. *Id*. at 1253. *See also infra* note 70 and accompanying text.

30. *See Sega Enters. Ltd.*, 977 F.2d at 1524. In support for its finding for a "fair

Existing copyright doctrine is notoriously conflicted in attempts to distinguish protected expression from "ideas" that more traditionally fall within the scope of patent law. In *Baker v. Selden*, for example, the Supreme Court held that a book on bookkeeping was copyrightable but not the ideas of bookkeeping in the book or the functional blank forms within the book.[31] "There is a clear distinction between the book, as such, and the art which it is intended to illustrate."[32] This notion of separating protected expression from ideas is known as the "idea/expression dichotomy," discussed in detail infra.[33]

In *Whelan Assoc., Inc. v. Jaslow Dental Lab., Inc.,* on the other hand, the court held that some structural elements of a software program designed to manage a dental laboratory were comparable to the structural elements of general literary works and broadened copyright protection for what would appear to be considered uncopyrightable ideas under *Selden.*[34] As illustrated further in this paper, it can be an extremely complex, costly, and subjective endeavor to separate the "process" or "procedure" from software and its form of expression.

D. Protection of Software through Technology

Protection from software piracy is also available through existing

use" defense, the court in *Sega* acknowledged the substantially functional nature of the software code and its aspects that were were not copyrightable. *Id*. at 1526. *See also supra* notes 21-23.

31. 101 U.S. 99 (1879). In *Baker*, the plaintiff, Selden, authored books on methods of bookkeeping, which contained numerous blank forms for practicing the methods. *Id*. at 99-100. The defendant, Baker, practiced methods similar to those described by Selden and used forms with formatting (i.e. lines and headings) similar to those published in Selden's books to which Selden claimed copyright protection. *Id*. at 100-01. The Court reasoned that while Selden's description of the system of bookkeeping may be copyrightable, the methods or systems themselves were not within the purview of copyright, and the rights to exclude their use were governed exclusively under patent law. *Id*. at 102-03. Since the forms published by Selden were an integral aspect of practicing the uncopyrightable methods, the forms in general were also not copyrightable. *Id*.

32. *Id*. at 102.

33. *Baker* offers an example of where writings or similar works of authorship (i.e. forms), that might normally fall under copyrightable subject matter, are precluded from full copyright protection because the published work itself becomes part of an non-copyrightable idea (or merges with it). *See supra* notes 31-32 and accompanying text; *see also* notes 50-57 and accompanying text (for a further discussion of *Baker* in relation to the idea/expression dichotomy).

34. 797 F.2d 1222, 1236 (3d. Cir. 1986).

software technology itself, aided in part by the Internet. For example, many mass distributed software programs now require a user to go through an active registration process before being able to use the software.[35] Some vendors also require that each licensed installation be inextricably tied with the unique processor identification number that every computer includes. Circumvention of these protections can incur criminal and civil liability under the Digital Millennium Copyright Act ("DMCA").[36] As a result, concerns have been expressed with regard to these practices under privacy and convenience issues.[37]

Physical solutions such as hardlock keys or "dongles" which use encryption technology to prevent unauthorized use of software are also available.[38] Such devices must generally be inserted into a port of the computer before the software will operate.[39] The typical concerns with regards to these forms of protection are convenience related (e.g., in the event a key is lost or broken).[40] Circumvention of hardware-based protections can also create liability under the DMCA.[41]

35. *See How to activate Windows XP,* Sep. 2006, *at* http://support.microsoft.com/default.aspx?scid=kb;en-us;307890 (last visited December 30, 2006) (explaining why an "active" registration process over the Internet or phone is employed during installation to prevent unlawful distribution).

36. 17 U.S.C. § 1201(a)(1)(A) (2005) (Providing that "[n]o person shall circumvent a technological measure that effectively controls access to a work protected [under Copyright] under this title."). *See also* RealNetworks, Inc. v. Streambox, Inc., No. 2:99CV02070, 2000 WL 127311, at *7-8 (W.D. Wash. Jan. 18, 2000) (holding that a device which circumvented code protecting streaming content from copying was in violation of the DMCA).

37. *See Learning Cyberlaw in Cyberspace*, http://www.cyberspacelaw.org/ (last visited Dec. 30, 2006) (providing content and links in relation to the legal issues surrounding online privacy).

38. *See* STEVEN M. KAPLAN, WILEY ELECTRICAL & ELECTRONICS ENGINEERING DICTIONARY 207 (2004) (defining dongle as "A software copy-protection device which usually plugs into a parallel port").

39. *See id.*

40. *Cf. Vault Corp. v. Quaid Software Ltd.*, 847 F.2d 255 (5th Cir. 1988) (holding (pre-DMCA) that software allowing users to operate software program without the original diskette in the computer did not constitute copyright infringement because users had a right under copyright law to make archival copies).

41. *See* Universal Studios, Inc. v. Corley, 273 F.3d 429 (2d Cir. 2001) (holding that the posting of code for circumventing DVD encryption technology was in violation of the DMCA).

IV. Perspective of Policies behind Copyright Law

There are differing and perhaps changing policies behind existing copyright law. The copyright clause of the Constitution expresses that authors should have "the exclusive Right to their...Writings" to "promote the Progress of Science."[42] Some argue that the major purpose behind copyright law is or should be a "quid pro quo" or bargain arrangement between authors and the public where the author gets a limited exclusive right to her writings in exchange for her disclosure of ideas.[43] According to the Supreme Court in *Baker*, "the very object of publishing a book on science or the useful art is to communicate to the world the useful knowledge which it contains."[44]

Even the Supreme Court, however, has held recently that disclosure may be merely an "objective" rather than an obligation of the author, and thus, having an "incentive" for creating "works of art" is the primary goal behind copyright.[45] Moreover, many admit that what was meant by "to promote the Progress of Science and useful Arts" is not fully settled.[46] The pre-cursor *Statute of Anne* appears principally concerned with maintaining control over distribution in order to provide reprieve for discouraged authors whose works were methodically copied without permission.[47] Furthermore, many musical or other artistic works arguably do not express much in the way of ideas and are not technically "useful" or considered "progress,"[48] while present day copyright laws appear to favor artist's

42. U.S. CONST. art. I, § 8, cl. 8.

43. *See* Samuelson, *supra* note 11, at 710-12 (citing that, prior to the advent of machine code, the purpose of publication was generally to communicate the knowledge contained within the publication).

44. 101 U.S. at 103.

45. *See* Eldred v. Ashcroft, 537 U.S. 186, 211-13 (2003). In allowing for the extension of term limits to copyrights, the Court explains that the judiciary should defer to Congress on how best to promote the "progress" of science within the meaning of the Constitution. *Id*.

46. U.S. CONST. art. I, § 8, cl. 8. *See Eldred*, 537 U.S. at 211-13 (determining that what promotes the "progress of the useful arts" may be a changing standard).

47. 8 Ann., c. 19 (1710) (Eng.). This statute is seen as the founding piece of copyright legislation that led to our own laws. *See* COPYRIGHT LAW 16-18 (Craig Joyce, et al. ed., 6th ed., 2003). The *Statute of Anne* is one of the first known copyright acts to grant rights in authors. *Id*. Prior to the *Statute of Anne*, rights had generally been vested in printers and booksellers. *Id*. The writers of the *Statute of Anne* at least purported to be concerned with "the very great detriment" that unauthorized copying causes authors who "compose and write useful books," COPYRIGHT LAW (quoting 8 Ann., c. 19 (1710) (Eng.)).

48. U.S. CONST. art. I, § 8, cl. 8 (known as the patent and copyright clause).

individual rights rather than the public's interest in disclosure.[49]

A. The Idea/Expression Dichotomy

The primary problem for the courts applying copyright to forms of expression like software is the combined utilitarian and expressive nature of the work itself. As held in *Baker*, copyright protection is only extended to the expression of an idea but not to the idea itself.[50] The distinction between the idea and its expression is referred to commonly as the "idea/expression" dichotomy.[51] In *Baker*, the idea in contention was the use of forms in particular formats for bookkeeping.[52] "The copyright of a book on book-keeping cannot secure the exclusive right to make, sell, and use account books prepared upon the plan set forth in such book."[53] The defendant in *Baker*, however, may have been found liable for infringement for making exact duplicates of the forms from the pages of Selden's book if the idea behind them could be expressed in a number of varying ways.

When an uncopyrightable idea can only be expressed in a single or limited number of ways it is said to have "merged" with its expression.[54] In *Apple Computer, Inc. v. Microsoft Corp.*, many features or "ideas" found in Apple Computer's original Macintosh operating system, such as overlapping windows, were found to be ideas that had merged with their expression and, thus, could not be copyrighted.[55] As a result, Microsoft could incorporate these and

49. The provisions of the Copyright Act of 1909 only granted copyright protection upon publication of a work. *See* Copyright Act of 1909, Pub. L. No. 349, 35 Stat. 1075 (1909) (replaced by the Copyright Act of 1976). *See also* Joyce et al., supra note 47, at 20. In contrast, under the present laws, protection is granted immediately upon creation. *See id*. at 22.

50. 101 U.S. 99. *See also supra* notes 31-32 and accompanying text (introducing the facts of *Baker* and the concept behind the "idea/expression" dichotomy).

51. *See, e.g.*, 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2.03[D] (2006).

52. 101 U.S. 99.

53. *See id*. at 104.

54. *See* Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738 (9th Cir. 1971) (holding that the idea of a "jeweled bee pin" had merged with its expression and could not be copyrighted).

55. *See* Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006 (N.D. Cal. 1992) [hereinafter *Apple Computer I*], *aff'd in part,* 35 F.3d 1435, 1444 (9th Cir. 1994) [hereinafter *Apple Computer II*] (affirming the "merger" and "scènes à faire" doctrines and their application to certain desktop features, including those of Apple's windows-based desktop architecture). The "windows desktop" features

many other ideas used within Apple's Macintosh "desktop theme" in their new Windows operating system and avoid copyright liability.[56]

The idea/expression dichotomy is expressly codified in 17 U.S.C. § 102(b) of the Copyright Act for ideas that pertain to any "procedure, process, system, method of operation, concept, principle, or discovery..."[57] For example, the Court viewed the bookkeeping procedure of *Baker* as a method or system of organizing books, a category of subject matter that is expressly prohibited under the current 17 U.S.C. § 102(b).[58]

It is significant that computer programs are frequently identified as "systems," their routines as "procedures," and running programs or sub-programs as "processes."[59] As will be discussed further in this paper, the courts have struggled with finding a consistent way of applying the "idea/expression" and "merger" doctrines to software.

Another related line of defense against copyright infringement is the "scènes à faire" doctrine.[60] Where a work has elements that are incident to or are inevitable aspects of unprotected ideas, those elements also become excluded from protection.[61] This doctrine resulted generally from attempts to protect common aspects of fictional works, such as certain character types or themes in particular

that Apple claimed a copyright in were originally developed by engineers at Xerox's Palo Alto Research Center in the 1970s and further developed at Apple by several of the same engineers. *See Apple Computer I,* 799 F. Supp. at 1017-18. The subject matter which Apple was attempting to protect was not software code per se but aspects of the arrangement and operability of the operating system's graphical user interface (i.e. icons, menus, point-and-click operability). *See id*. at 1015-17. The court rejected an overall "look and feel" analysis toward the determination of the substantial similarity component of copyright infringement and instead independently analyzed specific elements of the interface and identified which were protected and which were not. *See id*.

56. *See id*. The decision in *Apple Computer I* (upheld in *Apple Computer II*) held that those elements of Apple's interface which were not otherwise licensed, including overlapping windows, icons, and menus, were not protectable pursuant to previously established copyright doctrines, including "merger," "scènes à faire," and/or because they lacked originality. *See Apple Computer I, 799F. Supp*. at 1027-41.

57. *See supra* note 29 and accompanying text (for further discussion of the legislative intent behind 17 U.S.C. § 102(b)).

58. 17 U.S.C. § 102(b); *Baker v. Selden*, 101 U.S. 99, 104 ("Selden, by his books, explained and described a peculiar system of book-keeping, and illustrated his method").

59. *See, e.g*., KAPLAN, *supra* note 38, at 824 (defining UNIX as a "popular multitasking operating system").

60. *See, e.g*., 4 NIMMER, *supra* note 51, § 13.03[B][4].

61. *Id*.

genres of literature and theater.[62] The doctrine has been extended to more utilitarian works such as architecture where certain types of features (e.g., flying buttresses) have become "inevitable" components of designs meant to embody particular styles.[63] Not surprisingly, defendants of copyright infringement suits in software cases now argue that many features or details of certain types of programs are incident to or inevitable to those programs and are thus not protected by copyright.[64]

The development of the "idea/expression" dichotomy and the related "merger" doctrine appear to be a reaction by the courts to keep copyright protection distinct and separate from the ideas and practices within science and technology, including that of patentable subject matter. The "methods of useful art have their final end in application and use...But as embodied and taught in a literary composition or book, their essence consists only in their statement. This alone is what is secured by copyright."[65] Little did the Court in *Baker* realize that even a "statement" (e.g., a line of computer code) could be considered part of a "final end in application and use."[66] Thus, the development of these doctrines leaves the door open for infringement of software that goes beyond literal infringement.

B. Overview of CONTU

Partly in response to the growing use of copyright for the protection of software in the 1960s and 1970s, Congress created the

62. *See* Cain v. Universal Pictures Co., 47 F. Supp. 1013, 1017 (S.D. Cal. 1942) (holding that small elements shared between church scenes in a movie and book were inherent to the genre of the scenes and were not infringing).

63. *See* Domingo Cambeiro P.C. v. Advent, Nos. 99-17057, 99-15893, 2000 U.S. App. LEXIS 3658, at *3 (9th Cir. 2000). In *Domingo Cambeiro*, the plaintiff originally submitted an architectural plan for a Las Vegas Casino with a New York City (NYC) theme, after which the defendants adopted a NYC theme that the plaintiff claimed infringed his copyright. *Id.* Although the court found that plaintiff's work did have copyrightable elements, those general elements which were inherent to most any NYC theme were not copyrightable pursuant to the scènes à faire doctrine. Id. at *4. According to the lower court's decision, granting one individual exclusive rights to all NYC-styled themes would be "abhorrent." *Id.*

64. *See Apple Computer II*, 35 F.3d at 1444 (determining that certain ways of performing functions on Apple's desktop architecture were indispensable to the idea and limited to only very limited forms of expression). In *Apple Computer I*, a "scènes à faire" table was submitted as evidence to demonstrate that certain features (e.g., overlapping windows, icons, menus) were common, inevitable aspects of most graphical user interfaces. 799 F. Supp. at 1024.

65. *Baker v. Selden*, 101 U.S. 99, 104.

66. *Id.*

National Commission on New Technological Uses of Copyrighted Works (CONTU). CONTU was organized in order to issue a report to qualify such protection under current doctrine and make recommendations for changes if necessary.[67] The CONTU commissioners recognized the growing importance of software and the need for protection in accordance with other works of intellectual property, stating that "copyright is likely to be increasingly important in protecting computer programs, particularly those of small entrepreneurs who create their works for individual consumers and who can neither afford nor properly use other forms of protection."[68]

The CONTU commissioners concluded that the protection of computer software under copyright comported with the intention of the 1909 and 1976 Acts in relation to protection of "literary works," insofar as the Acts represented the original expression of ideas.[69] In addressing the issues under section 102(b) and the "idea/expression" doctrine, the CONTU commissioners concluded that software copyrights would not grant monopolies to ideas because "[w]hen other language [code structure] *is* available, programmers are free to read copyrighted programs and use the ideas embodied in them in preparing their own works."[70] Part of their conclusion was supported by the following exchange in the report:

> Commissioner Miller: How many different ways are there to produce a program...?
> Dan McCracken: [* Vice President of the Association for Computing Machinery] An infinite number in principle, and in practice dozens, hundreds.[71]

Although the Commission acknowledged the increasing difficulties that will arise when attempting to distinguish whether a program is of an "infinite number of variations" of expression or a unique idea itself, the Commission appeared to simply pass this task onto the courts with fingers crossed. "Should a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case by case basis by the institution designed to

67. *See* NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS 1 (1979) [hereinafter CONTU FINAL REPORT], *available at* http://www.digital-law-online.info/CONTU/PDF/index.html (last visited Dec. 30, 2006).

68. *Id*. at 15.

69. *See id*. at 16.

70. *Id*. at 20.

71. *Id*. at 20.

make fine distinctions, the federal judiciary."[72] As will be seen later in this paper, in cases where an alleged infringer takes an existing work and makes minor changes, the Commission may have been overly optimistic about the judiciary's ability to consistently and fairly make such distinctions.

What may have provided impetus to the CONTU commissioners' recommendations was the uncertainty surrounding other available forms of protection at that time. Based on what they understood of present patent laws, the Commission did not hold out much hope that many computer programs as such could be patentable.[73] As we now know, under *State Street*, fairly broad protection under patent law is presently available.[74] Furthermore, the Commission's members also felt that trade secret law was both "inappropriate" and unfeasible given the nature and mass distribution of commercial software.[75] Trade secret law was deemed "inappropriate for protecting works that contain the secret and are designed to be widely distributed" because "[p]rotection is lost when the secret is disclosed..."[76]

Perhaps members of the Commission did not completely understand the difficulty and expense in reverse engineering a large and complex software program (e.g., Microsft Word) in order to gain sufficient understanding to reap much of the valuable and unique "expression" that the high level programming discloses. As for simple low-level programs embedded in products like digital timers, there is less likely to be any "unique" expression involved. Patent protection for software that closely mimics the hardwired predecessors would arguably be more appropriate. As for the "inappropriateness" of protecting secrets within mass distributed products, the purveyors of such products as Coca-Cola, Guinness, and others might disagree.[77] Although, unlike software, such tangible products are not easy to "literally" copy without knowledge of the trade secrets, this paper does not argue that protection from literal

72. *Id*. at 22.

73. *See* CONTU FINAL REPORT, *supra* note 67 at 17 (wherein the CONTU commissioners express doubt about whether a patent may ever be obtained on a computer program); Gottschalk v. Benson, 409 U.S. 63 (1972) (holding that the mere manipulation of data was not patentable subject matter despite plaintiff's argument that a method for converting from one decimal system to another was a sufficiently specific and pragmatic application).

74. *See State Street*, 149 F.3d 1368, 1373 (holding that the transformation of data representing "discrete dollar amounts…into a final share price, constitutes a practical application of a mathematical algorithm").

75. CONTU FINAL REPORT, *supra* note 67, at 17.

76. *Id*.

77. *Cf. id*.

copying of software should cease to continue.

V. Perspectives of the Policy behind Patent Law

The underlying difficulty with protecting software under copyright law is that the expression readily transforms into an embodiment that is a concrete, tangible, and useful process which typically falls within the exclusive province of patent law.[78] However, certain works of software may not be sufficiently "concrete" and "tangible" to be protected under patent.[79] Early attempts to patent software were blocked under the so-called "mental steps" doctrine and "business method" exception, holding that the mere manipulation of data is too abstract and akin to a mathematical algorithm.[80] Thus, an algorithm for general purpose calculations on a computer may not be specific enough.

Prior to *State Street*, the patentability of software was limited generally to those processes which resulted in "physical transformations."[81] The line between abstract and concrete was broadened in *State Street*, in which it was held that tying at least one sufficiently "specific" use or result to the software was enough to place it within patentable subject matter.[82] In *State Street*, the Court held that an algorithm transforming financial data into a particular financial plan was sufficiently practical.[83] Although the doctrine that emerged from *State Street* significantly opened patentable subject matter to many more computer related software applications, there is still some doubt over whether certain computer programs regarded as

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

79. *See generally* 1 DONALD S. CHISUM, CHISUM ON PATENTS § 1.03[6] (2006) (discussing the "mental steps" doctrine, a component of which is the notion that "a patentable process must be part of the 'useful arts,' the field of industrial technology as opposed to the 'liberal arts' or the social sciences").

80. *See Gottschalk v. Benson*, 409 U.S. 63, 68-70 (1972) (holding that converting binary numbers from one number system to another did not provide a sufficiently specific, practical use).

81. *See United States Patent and Trademark Office, Examination Guidelines for Computer-Related Inventions*, 61 Fed. Reg. 7478, 7483 (Feb. 28, 1996) ("to be statutory, a claimed computer-related process must either: (1) result in a physical transformation outside the computer for a . . . practical application . . . or . . . (2) be limited by the language in the claim to be a practical application. . . .").

82. *See State Street*, 149 F.3d 1368, 1373 (holding that algorithms are unpatentable unless applied in "useful" manner).

83. *Id*.

"general use" are patentable. Many such ideas may be left in an unprotected void if not found sufficiently "practical" for patent law or too "practical" for copyright law.

The inherent difficulty in accessing the underlying "ideas" embedded within distributed software programs makes proving or disproving novelty or nonobviousness during prosecution and infringement extremely difficult. For instance, a program that processes data about investment portfolios could use an almost infinite number of potential algorithms in any number of ways. However, unless information about the high level source code were divulged, the program's underlying algorithms would be nearly impossible to determine. Therefore, unless the distributed machine code is exactly the same as the infringing product, it is generally impossible to prove that the underlying method is being infringed without forcing the infringer to disclose their source code. This problem makes patenting certain kinds of "mental steps" (e.g., data translation) software highly impractical. Another side-effect of this "secrecy" problem has been the issuance of many software patents with dubious claims to novelty or nonobviousness.[84]

A number of defenses against dubious software patents are available but may not be satisfactorily effective. Third parties can attempt to instigate reexamination proceedings with the PTO by submitting prior art evidence at any time after issuance.[85] The proceedings, however, are *ex parte* and an unfavorable decision for an alleged infringer estops them from challenging the patent's validity in subsequent court proceedings.[86] Congress also partially reacted to the doubtful validity of some software patents by providing a special defense for business method claims under 35 U.S.C. § 273(b)(1), which provides an affirmative defense where the infringer (1) made a good faith reduction to practice and (2) commercially used by the infringer within a year prior to patent's filing date.[87] A

84. *See* Richard S. Gruner, *Intangible Inventions: Patentable Subject Matter for an Information Age*, 35 LOY. L.A. L. REV. 355, 367-368 (2002) (arguing that, while "intangible" inventions such as software presently lack adequate records of public disclosure, a steady compilation of new patents will eventually address this problem). *See generally* Robert P. Merges, *As Many as Six Impossible Patents Before Breakfast: Property Rights for Business Concepts and Patent System Reform*, 14 BERKELEY TECH. L.J. 577 (1999) (discussing other challenges posed in examining business method patents).

85. 35 U.S.C. § 302.

86. 35 U.S.C. § 305. *See also* KIMBERLY A. MOORE ET AL., PATENT LITIGATION & STRATEGY 768-71 (2d ed. 2003) (discussing the advantages and disadvantages of reexamination for patentees and accused infringers).

87. 35 U.S.C. § 273(b)(1).

defense of prior "commercial use," however, arguably does not provide much additional ammunition beyond the already existing "on sale" bar in 35 USC § 102(b).[88]

VI. Disseminating Software's Ideas And Expression

Under either patent or copyright, the challenge is to extract those aspects of software that fall within or outside each area of law. The extraction becomes especially delicate with respect to the effort to distinguish copyrightable expression from a potentially patentable process.

Under copyright, the tests that have been established for separating the "ideas" behind software from their expression have met with mixed and seemingly incongruent results. These tests fall under the more general "substantial similarity" prong for establishing copyright infringement. Either before or after an objective substantial similarity comparison is made, the court decides which aspects of the software are protectable and which are not.[89] If there is nothing copyrightable to compare, then there is no infringement.[90]

In *Apple Computer, Inc. v. Microsoft Corp.*, the District Court for the Northern District of California employed a two-part test where the expressions of the works in contention are objectively compared for similarities first.[91] Experts may take part in the comparison for establishing what criteria will determine their similarity. [92] If the expressions are found to be "substantially similar," the compared components are then judged as to whether they qualify "as an expression of an idea [and not] an idea itself."[93]

88. *See* 35 U.S.C. § 102(b) (which bars patentability to inventions that have been "in public use or on sale" more than one year prior to filing for patent). *See also* Egbert v. Lippmann, 104 U.S. 333 (1881) (commercial use or sale has been determined to be "public use" within the meaning of the patent statutes).

89. *See Apple Computer I*, 799 F. Supp 1006, 1020-21 (N.D. Cal. 1992) (if "substantial similarity" is found under the two part test, then the court looks to whether the expression and ideas have merged and are, thus, unprotectable under copyright); Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706-12 (2d. Cir. 1992) (under a three part "substantial similarity" test - abstraction, filtration, comparison - the protected expression is "filtered" from the unprotected expression prior to "comparison" for similarity).

90. *See Computer Assoc. Int'l*, 982 F.2d at 706 (holding that the merger of expression and idea precludes copyright protection of the expression).

91. *See Apple Computer I*, 799 F. Supp at 1020.

92. *Id.*

93. *Id.* (quoting Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173 (9th Cir. 1989)).

In *Computer Assocs. Int'l*, the court adopted a three step "separation" process consisting of (1) an "abstraction" analysis, (2) a "filtration" step, and (3) a comparison test.[94] The "abstraction" process consists of identifying the various levels of the program's abstraction, the highest level consisting of the primary functions of the program (e.g., determining an ideal portfolio of mutual funds) which can be broken down into sub-levels of modules and other functions and further broken down into subroutines and ultimately individual lines of code.[95] The filtration process advanced by the court determines whether the components at each level are "ideas" or "expressions" that are necessarily incident to the ideas ("merger"), or are otherwise unprotected (e.g., fall within the public domain). [96] Once the filtration process is complete, a collection of elements that are deemed protected expression are left over and compared with the corresponding elements, if any, of the alleged infringing work.[97] Although the test adopted under *Computer Assocs. Int'l* appears to provide more detailed guidelines than the *Apple Computer, Inc.* decision, a critical result of each test is that the courts will ultimately decide whether a work of software represents an uncopyrightable idea or one of many copyrightable ways to express the idea.

The task of separating idea from expression is less "objective" or obvious than the courts seem to acknowledge. One could argue that one of many ways of expressing a particular idea is also a distinct uncopyrightable idea. The problem subjects itself to a great deal of philosophical debate about the nature of elements under the "idea/expression" dichotomy and promotes a potential "battle of the experts" scenario to determine what the "objective" position of a reasonable software engineer would be. Furthermore, the "public domain" exception is also difficult to establish for the same reasons that patent examiners and litigants have when disputing novelty and nonobviousness.[98]

The outcome of the test as it was applied in *Computer Assocs. Int'l* provides much argument for why the test is suspect. In *Computer Assocs. Int'l*, the alleged infringer gained access to confidential software of the plaintiff and apparently created its own software by making small changes to the original in order to avoid literal

94. *See Computer Assocs. Int'l*, 982 F.2d at 706.

95. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707 (2d. Cir. 1992).

96. *See id.* at 707-10.

97. *See id.* at 710.

98. *See supra* note 84 and accompanying text (discussing challenges to properly examining software patent applications).

infringement.[99] The court found that the higher-level ideas or structure of the software could not be protected and that, because none of the modified lines of code were "identical" to the original software, there was no infringement.[100] Does this mean that only lines of code virtually identical are "similar" under copyright law? What does this say about the likelihood of getting copyright protection for software not literally infringed upon?[101]

VII. The Copyrightability of Machine Code

Although literal copying of machine code has been protected under copyright, such protection has come under attack on the grounds that machine code does not qualify as human readable "speech" or an expression in the traditional manner of a literary work, musical score, or even high-level software code.[102] The "expression" of higher-level code meant to be understood by software engineers cannot be recognized, without extreme difficulty, in the compiled machine code that is distributed to end users. In its most basic form, machine code appears as a series of ones (1's) and zeroes (0's) and, even with the aid of a decompiler, to most it does not appear to be more than a seemingly random sequence of machine commands. Since there is little human readable "expression" present in the machine code itself, some argue that the code does not fit within the purview of copyright law.[103] Others argue that works which convey even the slightest bit of expression and information, even if only understandable by a handful of experts, falls within copyright's protections as a "literary"

99. *Computer Assocs. Int'l,* 982 F.2d at 698-701.

100. *Id.* at 714-15 (agreeing with the lower court's finding of there being no protectable expression in the structure of the code and determining that since "virtually no line of code" remained identical, there was no "similarity" between the works).

101. *See generally* Steven R. Englund, *Idea, Process, or Protected Expression? Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 886 (1990) (providing additional analysis of the protection afforded the structure and non-literal elements of computer programs).

102. *See* Samuelson, *supra* note 11, at 704 (arguing that expression has traditionally only been extended copyright protection if it is "human readable" either directly or with the aid of a machine (e.g., a book, recording)). *But see* Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 982 (1993) (arguing that machine code is merely a new medium of expression, discounting argument that its direct unreadability should preclude copyright protection).

103. *See* Samuelson, *supra* note 11, at 704.

work.[104]

Others contend that machine code, because of its utilitarian nature and generally unreadable format, should be limited in protection to expression produced by its execution, similar to a phonorecord or compact disc.[105] The drafters of the CONTU Report felt, however, that such a limitation would leave a great deal of software code (i.e. that which has little or no audio/visual output) unprotected from copying.[106] Furthermore, some argue that, because machine code is so far removed from the expression of the original source code, and not in accordance with a doctrine of complete disclosure, machine code does not deserve the same copyright protection as source code.[107]

VIII. European Copyright and Patent Protection for Software

Perhaps in response to the gradually more liberal and conflicting treatment of the patentability of software in the U.S., the European Patent Convention (EPC), an agreement binding on many European states, explicitly restricted the scope of software patents in Article 52 of that act.[108] Inventions not "susceptible of industrial application", which do not provide a concrete "technical result," are not considered patentable under European Patent Office (EPO) rules.[109] Examples

104. *See* Miller, *supra* note 102, at 989; Universal City Studios, Inc. v. Corley, 273 F.3d 429, 448 (2d Cir. 2001) ("Instructions such as computer code, which are intended to be executable by a computer, will often convey information capable of comprehension and assessment by a human being.").

105. *See* CONTU FINAL REPORT, *supra* note 67, at 27 (concurring opinion of Commissioner Nimmer, arguing unsuccessfully that machine code should only be protected under copyright to the extent of its expressive output when executed (e.g., windows, screens, sounds, graphics, etc…).

106. *See* CONTU FINAL REPORT, *supra* note 67 at 52-53 (Commissioner's majority opinion arguing against Commissioner Nimmer's suggestion, *supra* note 105, indicating that a significant amount of utilitarian-based non-expressive machine code would unjustifiably be left unprotected).

107. *See* Samuelson, *supra* note 11, at 705. *See also supra* note 41 and accompanying text (justifying that only expression which discloses the ideas embodied within it, which generally excludes machine code, should be afforded copyright protection).

108. *See* Convention on the Grant of European Patents art. 52 § 2(c), Oct. 5, 1973, 1065 U.N.T.S. 199, *available at* http://www.european-patent-office.org/legal/epc/ar52.htm, (as amended in 1993) [hereinafter referred to as the *EPC*] (wherein "programs for computers" "shall not be regarded as inventions.").

109. *See id.* *See also* 7 MANUAL FOR THE HANDLING OF APPLICATIONS FOR PATENTS, DESIGNS AND TRADE MARKS THROUGHOUT THE WORLD, European Patent Convention at 9-10 (Arnold Siedsma Ed. 2005) [hereinafter MANUAL FOR

of unpatentable subject matter include "schemes, rules and methods for performing mental acts, playing games or doing business, and *programs for computers.*"[110] Although the language sounds fairly resolute and still holds true for most software "as such," it was first indicated in an amendment of the EPC in 1985 that a program in combination with a machine may be patentable if the program causes the machine to operate in a new "technical" way or produces a "technical effect."[111]

Despite the explicit limit to software patentability, I.B.M. Corporation (IBM) attempted to argue[112] that a blanket exception to software technology is counter to the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS), which states that patents should be available in all fields of technology "capable of industrial application."[113] Although the EPO did not agree that they were bound by TRIPS, the result of the decision may be that European patents are available for some types of software "as such." The European system appears to have accepted the patentability of software with, at minimum, the potential for a "technical character."[114]

Meanwhile, attempts to legislate changes have been underway in order to revamp and broaden laws governing software patents under the European Union. These attempts, however, have met with a great deal of disagreement and consternation among some members.[115] A European Council Directive attempting to better define, harmonize, and generally broaden rights relating to software was soundly

THE HANDLING OF APPLICATIONS].

110. *EPC*, *supra* note 108, art. 52 § 2.

111. 7 MANUAL FOR THE HANDLING OF APPLICATIONS, *supra* note 109, at 9.

112. *See* Computer Program Product/IBM, T 1173/97 -3.5.1 §§ 2.1-2.5 (EPO Board of Appeals July 1, 1998) *available at* http://swpat.ffii.org/papri/epo-t971173/index.en.html (last visited Jan. 3, 2007) (in which the Board of Appeals acknowledges the significance of I.B.M.'s argument that under Article 27(1) of TRIPS (infra note 113), "patents shall be available for any inventions, whether products or processes.")

113. Agreement On Trade-Related Aspects Of Intellectual Property Rights, art. 27(1), Apr. 15, 1994, Marrakesh Agreement Establishing the World Trade Organization, Annex 1C, Legal Instruments – Results of the Uruguay Round, 33 I.L.M. 81, 93-94 [hereinafter TRIPS].

114. *See* Computer Program Product/IBM, T 1137/97 -3.5.1 § 5.1; *see generally* ROBERT PATRICK MERGES & JOHN FITZGERALD DUFFY, PATENT LAW AND POLICY: CASES AND MATERIAL 186-196 (3d ed. 2002) (discussing the exclusion of software patents under European Patent laws).

115. Jim Rapoza, *Poland's Stand Against European Patents Was Heroic*, Eweek News and Reviews (February 14, 2005) *available at* http://www.eweek.com/article2/0,1895,1761742,00.asp (last visited Nov. 8, 2006).

defeated.[116] On the other hand, another legislative initiative is underway to integrate and consolidate litigation relating to the validity and infringement of European patents,[117] thus giving greater strength to the EPO's relatively broad decisions regarding the patentability of software.[118]

Under copyright law, Europe treats software similarly to that of literary works and, as in the U.S., also does not appear to thoroughly address the "idea/expression" dichotomy.[119] Although infringement includes "permanent or temporary total or partial reproduction of the program by any means in any form" and "translation, adaptation, arrangement or any other alteration,"[120] a European Council Directive on the copyrightability of computer programs indicates that the underlying ideas are not to be protected by copyright.[121] When is an "adaptation" or "translation" of a computer program merely a copy of an underlying idea rather than the program itself?[122]

Prior to directives by the European Council, some member countries adopted substantially different copyright protection for software. In Germany between 1986 and 1993, for example, only those programs based on a personal intellectual creation exceeding that of the "average programmer" were considered protectable.[123]

116. Press Release, European Parliament, Software patents: the 'historic vote' in the European Parliament brings the battle to an end (Sept. 7, 2005), *available at* http://www.europarl.europa.eu/news/public/focus_page/057-1002-255-09-37-909-20050819FCS01001-12-09-2005-2005/default_en.htm (last visited November 18, 2006).

117. *See* European Patent Office, Legislative Initiatives in European Patent Law http://patlaw-reform.european-patent-office.org/epla/ (last visited Jan. 3, 2007) (proposing a system of uniform rules of procedure and empowerment of a common appeals court similar to that of the Court of Appeals for the Federal Circuit in the U.S.).

118. *See supra* notes 112-114 and accompanying text. Presently, patent litigation is carried out within the court systems of individual member countries, among which there may be varying degrees of acceptance to software patents. *See* supra note 115.

119. *See generally* MANUAL FOR THE HANDLING OF APPLICATIONS, *supra* note 109, Germany at 40.

120. *See* Council Directive 91/250, Legal Protection of Computer Programs, 1991 O.J. (L 122) 42 art. 4 ¶ 2, *available at* http://www.wipo.int/clea/docs_new/en/eu/eu020en.html# (last visited May 25, 2005) [hereinafter EC Directive].

121. *Id.*, art. 1 ¶ 2 ("Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive").

122. *See generally supra* notes 50-66 and accompanying text (describing the subjective issues encountered when attempting to separate idea from expression).

123. *See* MANUAL FOR THE HANDLING OF APPLICATIONS, *supra* note 109, Germany at 40.

Determinations about what constitutes expression exceeding that of an average programmer could likely lead to very subjective and arbitrary decisions, not unlike those that demarcate ideas, expressions, and their merger.

IX. Sui Generis Protection for Functional/Expressive IP

Perhaps as a deliberate effort to avoid the problems of extending copyright protection to highly functional objects or devices, several "carve out" statutes have been enacted for the special protection of certain intellectual property that shares both functional and expressive characteristics. These "carve-outs" include the Semiconductor Chip Protection Act (SCPA) of 1984,[124] the Vessel Hull Design Protection Act (VHDP) of 1998,[125] and protection for Design Patents under the Patent Act.[126] Although TRIPS has expressly mandated that member countries treat computer programs as copyrightable literary works in accordance with the Berne Convention,[127] the following examples of "carve outs" for other forms of functional/expressive IP could be useful if protection of computer programs in the U.S. and abroad is reconfigured to address the limitations of traditional copyright law.

A. Semiconductor Chip Protection Act

The SCPA protects masks, or electronic maps, of semiconductor chip designs from unauthorized duplication.[128] The major impetus for semiconductor design protection was that semiconductor designs, like software, are relatively easy to transform from expression into useful embodiments[129] and they represent a significant and expensive

124. 17 U.S.C. §§ 901-14 (2000).

125. *See* Digital Millennium Copyright Act § 504(b), Pub. L. No. 105-304, 112 Stat. 2860, 2905 (1998) (codified at 17 U.S.C. §§ 1301-1332)) [hereinafter DMCA].

126. 35 U.S.C. § 171.

127. TRIPS, *supra* note 113, art. 10 (setting forth that Computer programs, including machine object code, are to be protected as literary works under the Berne Convention). *Id*., art. 9, however, also specifies that copyright protection shall not extend to "ideas, procedures, methods of operation or mathematical concepts as such."

128. 17 U.S.C. § 902(a)(1) (defining the subject matter of the SCPA as a "mask work fixed in a semiconductor chip product").

129. *See generally* SOC Central, http://www.soccentral.com (providing resources, articles, and discussion forums relating to the art of software aided electronic circuit architecture and design).

amount of intellectual effort.[130]

To accommodate the competing aspects of expression and utility in semiconductors, various parts of the protective scheme draw in part from copyright and in part from patent law. Unlike copyright, which confers protection upon fixation, protection for semiconductors under the SCPA begins with either commercial exploitation or registration of the design.[131] Unlike patent and like copyright, there is no stringent examination process regarding novelty or originality. Registration acts as "prima facie evidence of the facts stated in the certificate" but does not require overcoming the "clear and convincing" burden found in patent invalidity claims.[132] Although the exclusive right to reproduction by "optical, electronic, or any other means"[133] is reminiscent of copyright-like protection[134], the duration of protection for mask works (ten years) is closer in nature to utility patent law (twenty years) from the filing date) and even closer to design patent law (fourteen years).[135] By limiting the duration of protection and requirements for registration as compared to utility patents, while extending copyright-like protection to expressions of the design, the SCPA strikes a balance between patent-like and copyright-like protection.

B. Design Patents

Design patents also offer protection for works that have elements of both patentable utility and copyrightable expression. A design patent offers protection to an "ornamental design for an article of manufacture."[136] Unlike the traditional utility patent, a design patent does not (and cannot) protect the function or utility of a machine.[137]

130. *See* Brooktree Corp. v. Advanced Micro Devices, Inc., 977 F.2d 1555, 1563 (Fed. Cir. 1992) (stating that the SCPA is uniquely designed "to achieve appropriate protection for original designs while meeting the competitive needs of the industry and serving the public interest").

131. 17 U.S.C. § 101 ("A [copyrighted] work is created when it is fixed…"). *Id.* § 904.

132. *See* 17 U.S.C. § 908(f). *See also* TypeRight Keyboard Corp. v. Microsoft Corp., 374 F.3d 1151, 1157 (Fed. Cir. 2004) (reaffirming that proving the invalidity of patents is presently subject to a "clear and convincing" standard).

133. 17 U.S.C. § 905(1). *See also id.* § 101 (as amended May 2003) (providing the definition of "copies" under U.S. Copyright law as "material objects…from which the work can be perceived, produced, or otherwise communicated…").

134. *See* 17 U.S.C. § 302.

135. *See id.* §§ 904-905; *Id.* § 154.

136. 35 U.S.C. § 171.

137. *See* UNITED STATES PATENT & TRADEMARK OFFICE, MANUAL OF PATENT

It also cannot protect mere articles of expression (e.g., pictures).[138] Protection extends, rather, to a novel appearance or non-functional ornamentation (e.g., shape) incorporated into an "article of manufacture" (e.g., a chair, car body, etc..).[139] Like utility patents, design patent applications must undergo a rigorous examination process[140] prior to issuance although the term of protection is typically three to six years shorter (fourteen years) than that of a utility patent,[141] Should a similar type protection be extended to mixtures of expression and function within software? Would this resolve the problem of providing reliable protection for novel software user-interfaces that may not squarely fit under either patent or copyright law?[142]

C. Vessel Hull Design Protection

Another extension of intellectual property which gives protection to works that straddle the line between expression and function is the Vessel Hull Design Protection Act (VHDPA) portion of the Digital Millenium Copyright Act.[143] The VHDPA, similar in scope to the Semiconductor Chip Protection Act, offers protection to original vessel hull designs.[144] Like the SCPA, the VHDA does not require

EXAMINING PROCEDURE (8th ed., Aug. 2006 rev.) § 1504.01(c) *available at* http://www.uspto.gov/web/offices/pac/mpep/mpep.htm [hereinafter MPEP] (in the section entitled "Functionality vs. Ornamentality," providing that "To be patentable, a 'primarily functional invention is not patentable' as a design" (quoting Norco Products, Inc. v. Mecca Dev., Inc., 617 F.Supp. 1079, 1080 (D. Conn. 1985))).

138. *See* MPEP, *supra* note 137 § 1504.01 (in the section entitled "Statutory Subject Matter for Designs," "A claim to a picture, print, impression, etc. per se, that is not applied to or embodied in an article of manufacture should be rejected . . .").

139. 35 U.S.C. 171. *See also* MPEP, *supra* note 137, § 1512 (in the section entitled "I. Design Patent/Copyright Overlap", indicating that an ornamental design may be simultaneously copyrighted and protected by design patent).

140. *See* MPEP, *supra* note 137, § 1504 (in the introductory section entitled "Examination," providing that novelty, nonobviousness, and ornamentality are prerequisites to patenting).

141. 35 U.S.C. 173; *see also* MPEP, *supra* note 137, § 1502.01(A) (providing that the term of a design patent is 14 years from the date of grant, rather than the 20 years afforded to utility patents).

142. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995) (rejecting a program menu-hiearchy as non-copyrightable subject matter on grounds that it was a functional aspect of the program).

143. *See* DMCA, *supra* note 125.

144. 17 U.S.C. § 1301(a)(1)-(2).

the rigorous examination process of patents and also extends for 10 years.[145] Although novelty is not expressly required, protection is only valid for works that provide a "distinguishable variation over prior work."[146] Protection also requires that designs bear a proper notice, which hearkens back to old provisions under copyright laws.[147] Moreover, an infringer must actually have knowledge that a design is protected.[148] The VHDPA also expressly disallows overlapping protection of hull designs and design patents.[149]

In light of the plethora of designs in the field that are difficult to locate or discover, the VHDA includes several provisions making the misuse of undeserving rights less likely. First, protection under the act requires that the actual vessel hull incorporate the design.[150] Requiring an actual physical embodiment is arguably more severe than the enablement and utility requirements for patentability, which only require submission of a disclosure that teaches one "of ordinary skill in the art" how to make and use the invention.[151] Furthermore, an introduction of an earlier work by another serves as "prima facie evidence" of a lack of originality,[152] rather than the more onerous "presumption of validity" standard that alleged infringers of issued patents must overcome.[153] This provision essentially puts litigants on equal footing, giving the alleged infringer an opportunity to challenge claims of originality or novelty under a preponderance of evidence standard.

X. The Case For and Against Separate Protection For Software

Because of its uniquely expressive and utilitarian nature, along

145. *Id*. § 1305(1).

146. *Id*. § 1301(b)(1).

147. *See* i*d*. §§ 1306-1307. *See also* Copyright Act of 1909 § 10, ch. 391, 61 Stat. 656 (requiring the affixation of a notice of copyright on published works).

148. *See* 17 U.S.C. § 1309(c) (requiring knowledge by an alleged infringer that the work was protected under the Act). *See also* U.S. COPYRIGHT OFFICE & U.S. PATENT AND TRADEMARK OFFICE, THE VESSEL HULL DESIGN PROTECTION ACT: OVERVIEW AND ANALYSIS 5 (2003), *available at* http://www.copyright.gov/reports/vhdpa-report.pdf [hereinafter VHDPA: OVERVIEW AND ANALYSIS] (providing a general discussion of notice requirements).

149. 17 U.S.C. § 1329 (providing that the issuance of a design patent will terminate protection under the Vessel Hull Design Protection Act).

150. *See* VHDPA: OVERVIEW AND ANALYSIS, *supra* note 148, at 3.

151. 35 U.S.C. 112 ¶ 1. *See also* MPEP § 706.03(c) (discussing in detail the examination guidelines pertaining to the requirement of an enabling disclosure).

152. 17 U.S.C. § 1309(f).

153. *Id*. § 282.

with its importance in business, technology, and daily living, some propose that laws distinct from traditional IP law be created to specially govern software. Arguments regarding this topic typically revolve around the public's paramount interests in software, including the effects on innovation and the consumer, and whether continuing to adapt current law to software would adversely effect the protection of other forms of intellectual property.

Some argue that the uncertainties in protecting software through copyright, discussed herein, have had or will have the effect of discouraging innovation.[154] In *Questioning the Necessity of Copyright Protection for Software Interfaces*, Matthew P. Larvick asserts that software improves most effectively in small increments over existing technology.[155] If these small changes routinely infringe upon the prior technology and are subsequently blocked, Larvick argues that innovation will be greatly harmed.[156] In Four Reasons and a Paradox: *The Manifest Superiority of Copyright Over Sui Generis Protection*, Jane C. Ginsburg counters that the industry is presently thriving despite many years of fairly standard worldwide copyright protection for software.[157] Ginsburg appears to contend that "if it ain't broke, don't fix it."

154. *See* David M. Maiorana, Comment, *Priviliged Use: Has Judge Boudin Suggested a Viable Means of Copyright Protection for the Non-Literal Aspects of Computer Software in Lotus Development Corp. v. Borland International?*, 46 AM. U. L. REV 149, 169-70 (1996) (discussing various arguments by commentators surrounding how the uncertainty of copyright protection for non-literal elements instills a fear of infringement in would-be developers of competing/similar products).

155. Matthew P. Larvick, *Questioning the Necessiry of Copyrights Protection for Softward Interfaces*, 1994 U. ILL. L. REV. 187, 202 (1994) (citing examples of how certain major software products such as Lotus 1-2-3 and Apple's famous interface of windows and icons grew out of similar prior competing products). Much of what is presently incorporated into Microsoft Windows grew out of Apple's designs. *See supra* note 55 and accompanying text.

156. The failure of IBM's OS/2 operating system may be attributed to developing the product from "scratch" and the inability of its developers to learn from and avoid many of the mistakes and missteps of its predecessors. *See Larvick*, *supra* note 155, at 202.

157. *See* Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559-60 (1994) (noting that most industrialized countries have agreed to provide copyright protection to software in the same manner as "literary works" and arguing against a proposal for a new form of protection for software, claiming that objections about copyright's ability to protect software are overly pessimistic and premature and that the courts are equipped to work out a balanced approach within the scope of copyright). *See also supra* note 127 and accompanying text.

Other commentators also argue for maintaining protection of software under copyright but propose adding privileges and/or provisions to ensure the promotion of innovation. In *Copyright Protection For The Non-Literal Elements of Computer Programs: The Need for Compulsory Licensing*, Aram Dobalian favors the Second Circuit's abstraction-filtration-comparison test set forth in *Computer Associates,*[158] but believes that compulsory licensing of dominant products may be necessary to protect and encourage the incremental steps of innovation imperative to software development.[159] Others argue, however, that many problems, including price disputes, poor administration by the government, and difficulty in enforcement would be overly costly and make such a system unmanageable.[160]

In *Privileged Use: Has Judge Boudin Suggested a Viable Means of Copyright Protection for the Non-Literal Aspects of Computer Software in Lotus Development Corp. v. Borland International?*, David M. Maiorana takes Aram Dobalian's proposal a step further and suggests limited compulsory licensing for non-literal aspects of software. Reflecting the concept of "privileged use" by Judge Boudin in *Lotus Dev. Corp. v. Borland Int'l,*[161] Maiorana proposes a system where copying would be permissible if a royalty is paid and the copying is related to compatibility or improving upon existing technology.[162] While Mairona acknowledges that such a system could be difficult and expensive to implement,[163] Maiorana advocates

158. Aram Dobalian, Notes and Comments, *Copyright Protection for the Non-Literal Elements of Computer Programs: The Need for Compulsory Licensing*, 15 WHITTIER L. REV. 1019, 1073 (1994) (concluding that applying levels of abstraction to software provides the most balanced method of separating protectable expression and unprotected ideas).

159. *See id*. at 1068 (justifying the need for compulsory licensing due to monopolistic anti-trust problems (i.e. Microsoft) and the impracticality of and lack of incentives for large software vendors to license various components of their products).

160. *See* Maiorana, *supra* note 154, at 179 (raising common concerns regarding compulsory licensing, including how and by whom such a system would be implemented and the difficulty in enforcement).

161. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 821 (1st Cir. 1995) (Boudin, J. concurring). Judge Boudin expressed concern that users of programs would be locked into particular vendors because competitors would be barred by copyright from making compatible software. *See id*. Judge Boudin proposed that use for "compatibility" purposes should be an exception to liability on the condition of a royalty payment. *See id*.

162. *See* Maiorana, *supra* note 154, at 182-187 (offering justifications and a scheme for the implementation of "privileged" compulsory licensing).

163. *See id*. at 179-180 (acknowledging that determining whether a licensee is

that such a system would strike the appropriate balance between the interests of copyright owners and the software industry (and consumers) as a whole.[164]

Yet another model for protecting software, departing significantly from the previously discussed paradigms, is "open source" software.[165] Although "open source" software is available for anyone to use and modify, it comes with hidden costs. For example, almost all variants of "open source" licensing require distribution of the source code with commercially distributed copies of the programs.[166] The GNU General Public License ("GPL"), for example, allows competitors to then freely copy, improve, sell, and/or distribute the original software once they are licensed a copy.[167] Not included in the list of freedoms is the freedom to control distribution once a single copy has been licensed to another.[168] The proponents of "open source," despite the obvious hindrance to obtaining significant monetary rewards, contend that this model promotes innovation and eliminates the costs to consumers associated with mass-produced commercial software.[169] Although "open source" has not gained

motivated to improve technology or merely replicate an idea is subjective and difficult). Administering the licensing system would subsequently involve added delays and expense. *See id.*

164. *See* Maiorana, *supra* note 154, at 176, 188 (predicting that "privileged" use would promote compatibility and standardization among software programs, encourage competition and innovation, while still providing compensation to copyright owners).

165. "Open source" generally refers to publicly shared and distributed source code (which represents the high level instructions underlying software programs as generally discussed *supra* in part II of this paper (Overview of what constitutes software)). See http://directory.fsf.org for a list of some available open source software (last visited November 6, 2005).

166. See http://www.opensource.org/licenses/bsd-license.php for examples of "open source" licenses (last visited November 6, 2005). *See also* FSF – Frequently Asked Question about the GNU GPL, http://www.fsf.org/licenses/gpl-faq.html (last visited Jan. 3, 2007) (addressing common questions of software licenses promoted by the Free Software Foundation [FSF]).

167. *See* FSF – Frequently Asked Question about the GNU GPL, *supra* note 166 (FSF indicates that ". . . *if* you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the program's users, under the GPL. . . "). *See also* FSF – The Free Software Definition, http://www.fsf.org/licensing/essays/free-sw.html (last visited Jan. 3, 2007) (indicating that once a copy under the license is distributed, the recipient essentially has the same rights as the distributor to copy, sell, etc. . .).

168. *See* FSF – The Free Software Definition, *supra* note 167 (listing the freedoms of those possessing an FSF open-source license).

169. The FSF contends that the "open source" model promotes the sharing of and public disclosure of what would be otherwise hidden source code, the modification

substantial traction in the marketplace, noteworthy efforts backed by major vendors are currently underway.[170]

The uniqueness of software as an intellectual property not only creates uncertainty relating to copyright, but also clearly under patent law as well. In *Innovation and Its Discontents*, Adam B. Jaffe and Josh Lerner study how the uncertainty of protection for new fields of technology like software dissuades development and innovation in those fields.[171] Jaffe and Lerner contend that even while patent protection has become stronger in recent periods, the ability of the Patent Office to properly examine patent applications, particularly those relating to software, is seriously flawed and unreliable.[172] Jaffe and Lerner argue that the issuance of software patents of dubious validity inherently restricts the development of overlapping (or incremental) inventions.[173] Adding to the woes of would-be

and distribution of such software without the fear of copyright violation, and the development of a well-known "coherent" body of software which would eliminate the need to use proprietary software. *See* FSF – Copyleft: Pragmatic Idealism, http://www.fsf.org/licensing/essays/pragmatic.html (last visited Jan. 3, 2007).

170. Sun Microsystems has released a version of a JAVA software development platform under version 2 of the GPL (which allows other programs that are built over (or merely linked to) the JAVA libraries to be distributed under separate (non-GPL) licenses). *See* Martin LaMonica, *Newsmaker: Sun's open-source odyssey*, CNET.COM, July 6, 2006, http://news.com.com/Suns+open-source+odyssey/2008-7344_3-6090956.html?tag=st.rn (last visited Jan. 3, 2007). I.B.M. Corporation has also lent its support to other "open source" initiatives, including "open source" versions of JAVA competing with those of Sun Microsystems. *See* Martin LaMonica, *IBM cool to Sun's open-source Java plan*, CNET.COM, November 13, 2006, http://news.com.com/2061-10795_3-6134853.html (last visited Jan. 3, 2007). Sun Microsystems and I.B.M., nevertheless, could have a greater interest in profiting through the sales of their hardware systems rather than merely through the sales of JAVA products which run on their hardware.

171. *See generally* ADAM B. JAFFE & JOSH LERNER, INNOVATION AND ITS DISCONTENTS: HOW OUR BROKEN PATENT SYSTEM IS ENDANGERING INNOVATION AND PROGRESS, AND WHAT TO DO ABOUT IT (2004). *See also* Katherine Macklem, *Patent Predators*, MACLEANS, Mar. 28, 2005, *available at* http://www.macleans.ca/topstories/business/article.jsp?content=20050328_10 2766_102766 (last visited November 7, 2005).

172. *See* JAFFE & LERNER**,** *supra* note 171, at 200-202 (pointing out that U.S. Patent and Trademark Office ("Patent Office") examiners primarily rely on existing patents and published applications for their prior art searches and much of what is used or practiced in industry (and not patented) cannot be reliably accounted for during examination, thus resulting in many patents issued to technology already available to the public). *See also* Merges, *supra* note 84; Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495 (2001).

173. JAFFE & LERNER**,** *supra* note 171, at 59, 197 (discussing how large companies with significant resources frequently adopt the practice of creating a "thicket" of patents surrounding particular technologies, stifling development in

infringers in court is the fact that a patent is presumed to be valid, and can only be found invalid by "clear and convincing" evidence.[174]

XI. Conclusion and Proposed New Rules for the Protection of Software

The level of original work and ingenuity invested in software unarguably deserves protection under the same policy considerations as patent and copyright. Unchecked copying of software is at least as damaging to its owners as that of owners and authors of books and music. The policies under the Constitution for the promoting "useful arts," [175] including those shared in international laws and treaties, should clearly apply to the field of software engineering. However, what characterizes the ideas or expression within software does not fit entirely into either traditional patent or copyright doctrines. Rather than drastically changing the scope of traditional copyright or patent laws, the governing agencies can adopt specific provisions for software that are comparative to provisions for other untraditional forms of intellectual property.

Leaving the laws in their current state does not promote an optimum level of innovation. The present scope of offerings in major categories of software technology appear to be ever more dominated by fewer vendors.[176] The uncertainties in copyright and patent protection, including the application of the presently adopted vague and inconsistent tests discussed in this paper, are inherently unfair to developers and harmful to innovation.

Patching existing copyright law by granting "privileged" use or compulsory licensing would likely still result in debate about who, if anyone, owns any particular non-literal element of software and will simply shift these same issues to another government-administrated front. Under Judge Boudin's proposal, the issue of what constitutes privileged use (e.g., for improvement and/or compatibility) could potentially add significant unexplored complexity to infringement disputes.[177]

---

these areas).

174. *See* Kaufman Co. v. Lantech, Inc., 807 F.2d 970, 973-74 (Fed.Cir. 1986). *See also* 35 U.S.C. § 282 (codifying that the burden of persuasion in a patent invalidity claim rests with the alleged infringer).

175. U.S. CONST. art. I, § 8, cl. 8.

176. For instance, it is generally acknowledged that Microsoft Corp. presently dominates the market in operating system, word processing, spreadsheet, and browser software.

177. *See* notes 160-163 and accompanying text.

The "open source" paradigm solves the property/ownership aspects of software innovation by essentially eliminating them. The proponents of "open source" do not provide a reasonable explanation as to what incentives will replace monetary rewards other than when profits can be tied in with additional services or products. It is just as likely that businesses will be more inclined to avoid "open source" so as not to put their own innovations in jeopardy of being dedicated to the public without cost.

New laws particular to software, rather, should follow along the basic public policy prongs corresponding to aspects of copyright. These changes should also be guided by the successes or failures of previously adopted "carve outs" for non-traditional intellectual property. As these provisions are adopted, a corresponding withdrawal of the more controversial changes to copyright and patent law relating to software should occur. In conjunction with these changes in the U.S., international agreements would also need to be adjusted, particularly under TRIPS,[178] so that "carve outs" could take the place of general copyright protection for software.

This author proposes new provisions which expressly protect the work from literal copying and direct use of executable (including commercially distributed) programs just as under present copyright law. Protecting original program executables from distribution, literal copying, or direct use would be akin to the "distribution" and "reproduction" rights under copyright. Literal protection would also extend to de-minimis modifications. The term of protection for literal copying should similarly be extended to that of present copyright law. By this provision, as under the copyright provisions, the public would not be generally prohibited from practicing the ideas contained within the executable program. The public would also be protected, for similar policy reasons as those under copyright, by similar "fair use" and "first sale" provisions.

These new provisions would recognize that, although executable code is not a traditional form of human-readable "expression" within the meaning of copyright, the work invested into non-literal and easily copyable embodiments of a "process" is worthy of protection and that such protection should be of minimal burden to the public. Owners of these new rights would continue to have existing trade secret laws available to them.

The rights to cover the protection of non-literal novel software inventions should be treated and registered independently. Registration would require a description and drawings to support an

178. *See supra* note 127 and accompanying text.

original, novel, and nonobvious software design and also require a listing of the inventive steps (akin to patent claims), alternatively in a pseudo-code format. Valid registration would be subject to substantially all of the same standards as that of patents. Instead of the rigorous examination process under patents, however, the submission would be subject to a rudimentary examination to ensure it meets formal requirements. For example, the examination requirements could be similar to the requirements of a vessel hull design or semiconductor design submission. Similarly protection would last approximately ten years and successful enforcement would be subject to overcoming challenges to validity (similar to that of patents).

Infringement actions, however, would pit plaintiff and defendant on substantially equal footing with regard to both infringement *and* validity, with all parties being subject to the "preponderance of evidence" standard.[179] This proposal would stem the unreasonable presumption that issued software patents truly meet existing patentability standards and alleviate the growing problem of expensive and arguably unfair litigation involving undeserved patent grants.

Together with the proposed new rules, the recent doctrinal expansion of both patentability and copyrightability would be reversed so that both bodies of law would be more harmonious with their foundations and international counterparts. Copyright law would no longer extend to software *per se*, just as it does not to processes, machines, circuits. Thus, a "procedure" or "process...regardless of the form in which it is described" under 17 USC § 102(b) would be strictly construed and be more consistent with the Berne Convention provisions. In addition, business methods and software "as such" would no longer be considered patentable subject matter. Patent examiners would no longer be required to judge whether code was merely an unpatentable "manipulation of data" or series of "mental steps" as opposed to concrete, specific, and useful methods. Patents, however, would still extend to software components that operate in combination with sufficiently traditional tangible processes and machines, similar to European patents and to U.S. patents prior to *State Street*. United States patent law would thus revert to its prior requirements for concreteness, be more

179. *See supra* note 132 and accompanying text. In contrast to patent infringement, which is subject to a "preponderance of the evidence" standard, patent validity can only be successfully challenged with "clear and convincing evidence."

consistent with similarly accepted international standards for industrial or "technical effects" and stem the tide of abusive lawsuits over unworthy patent claims. Copyright law would no longer extend into the domain of patentable subject matter and would not be subject to the arguably impossible task of determining whether software is idea, expression, or both.