UIC John Marshall Journal of Information Technology & Privacy Law

Volume 15
Issue 4 *Journal of Computer & Information - Summer 1997*

Article 6

Summer 1997

The West German Smorgasbord Approach to Intellectual Property Protection of Computer Software, 15 J. Marshall J. Computer & Info. L. 883 (1997)

Larry N. Woodard

THE WEST GERMAN SMORGASBORD APPROACH TO INTELLECTUAL PROPERTY PROTECTION OF COMPUTER SOFTWARE

I. INTRODUCTION

When reflecting upon the unprecedented growth of the computer industry, consumers frequently ask themselves: *What will they think of next?* Unfortunately, this growth by the computer industry is unmatched by an equaled surge of new and efficient regulation of the industry's intellectual property rights for computer software.[1] As a result, members of the computer industry find themselves asking the lawmakers: *Why haven't they thought of anything?* If members of the industry attempt to protect their valuable intellectual property in the software, they are forced to gaze through the currently murky waters of patents, copyrights, trademarks, and trade secrets.[2] This confusion, caused by legal inaction, leads to inefficient extremes of underprotection[3] and overprotection[4] of intellectual property rights when applying copyright and patent law.[5]

---

1. Victoria Slind-Flor, *A New Idea in Software Protection: A 'Manifesto' Calls for a Unique Law That's Neither Copyright Nor Patent,* NAT'L L.J., Jan. 16, 1995, at A1, (quoting James H. Lows, Microsoft corporate attorney, as saying that the $13 billion computer software business would be potentially worth twice that if proper legal protection were provided).

2. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712 (2d Cir. 1992). The *Altai* court admitted outright that the "contours of copyright protection," the predominant method to protect computer-related material, are not completely clear. *Id.* The court hopes that future cases will better define the limits of protection. *Id.* Unfortunately, these hopes are thus far unfulfilled. *Id.*

3. Ian C. Ballon & Heather D. Rafter, *Computer Software Protection*, 431 PLI/PAT 81, 87 (1996) (citing the congress' classification of computer software as a "literary work" for copyright protection cannot adequately protect the utilitarian nature of computer software).

4. *Id.* at 106 (commenting that software patents which grant a twenty year monopoly of intellectual property protection may overprotect and therefore harm the software industry).

5. Lotus Dev. Corp. v. Borland Int'l, Inc. (Lotus II) 788 F. Supp. 78, 91 (D. Mass. 1992). The court, here, was one of the first to allude to the forthcoming debate over the adequacy of copyright protection for computer programs and the possibility of patent protection for software. *Id.*

Part of the lawmaker's dilemma is that the concept and definition of "software" is not easily related to the conceptual and definitional standards of current intellectual property protection.[6] Because software tends to act more and more like hardware, the functional aspect of software does not fit into the subject matter domain of copyright law,[7] nor does software easily satisfy the "usefulness" requirement of the Patent Act.[8] Nevertheless, by increasing the protection for the idea and functionality[9] of software, lawmakers have demonstrated their understanding that heightened levels of technology require heightened levels of protection.[10]

This new protection comes in the form of an animal never-before seen in the jungle of intellectual property law: the software patent.[11] Through the software patent, lawmakers attempt to solve the problem,

---

6. Pamela Samuelson, et al., *A Manifesto Concerning the Legal Protection of Computer Programs,* 94 COLUM. L. REV. 2308, 2310-24 (1994) [hereinafter *Manifesto*]. The *Manifesto* argues that computer software allows computers to behave like machines with certain functional and utilitarian aspects commonly demonstrated by inventions traditionally protected by patents. *Id.* at 2312. The *Manifesto* makes the astute observation that computer hardware is somewhat useless and non-functional without computer software and vice versa. *Id.* The software allows the computer to "behave," and this behavior, like other patentable machines, is where the value lies. *Id.* at 2323.

The nature of the computer industry makes a semantically clear definition of "hardware" and "software," as well as their characteristics, increasingly difficult to articulate. *Id.* Since patents traditionally protect hardware, and copyrights traditionally protect software, this bright-line distinction between hardware and software contributes to the problem of how the law should protect hardware and software, by either patent or copyright. *Id.* at 2324.

7. John A. Kidwell, *Software and Semiconductors: Why Are We Confused,* 70 MINN. L. REV. 533, 552 (1985) (noting that one of the difficult issues of protecting software by copyright is that software does not fit into the traditional "writing" definition of copyright law's scope).

8. David G. Luettgen, *Functional Usefulness vs. Communicative Usefulness: Thin Copyright Protection for the Nonliteral Elements of Computer Programs,* 4 TEX. INTELL. PROP. L.J. 233, 248-49 (1996) (explaining that software is not traditionally considered "useful" for the purposes of protection under the Patent Act).

9. 17 U.S.C. § 30(a), (b) (1988 & Supp. 1990). Copyright law does not protect any "idea, procedure, process, system, method of operation, concept, principle or discovery" of a protected work, while patent law does. *Id.* Traditionally, intellectual property that "functions" is protectable only with a patent. *See* Patent Act, 35 U.S.C. §§ 1-376 (1988). In the case of software, the software, for the most part, is protected by a copyright. However, software causes the functional behavior of a computer, protectable by a patent. This makes the software "functional." *Compare* 35 U.S.C. §§ 102-03 (1988) *with* 17 U.S.C. § 102(b) (1988 & Supp. 1990) (showing the difference in qualifications in obtaining a patent with the scope of intellectual property protected by a copyright).

10. Diamond v. Diehr, 450 U.S. 175, 184-86. The court realized that the increased use of computers in society and thus allowed computers and digital machines protectable under 35 U.S.C. § 102, the Patent Act. *Id.*

11. *Id.*

but they certainly do not resolve the problem.[12] The great optimism for the new software patent is met with paralleled pessimism, with arguments that the patent is overprotective of intellectual property and difficult to grant without the historical "prior art" used to issue patents.[13] Likewise, copyright protection cannot adequately protect the innately functional medium of computer software.[14] To date, legal scholars conclusively appear to agree to disagree: given the current level of technology and the functional aspect of software, the application of copyright law is obsolete and underprotective,[15] while on the other hand, software patents are difficult to issue and are inefficient.[16]

This comment focuses on the current gaps in copyright and patent protection for computer software products. First, this comment provides the necessary background of copyright and patent law as it pertains to computer products. In doing so, this comment briefly discusses the application of intellectual property law to current computer technology. Then, this comment analyzes the positive and negative aspects of copyright and patent law by demonstrating the advantages and the difficulties of applying copyrights and patents to computer software. Finally, utilizing the favorable aspects of both copyright and patent law in a "smorgasbord" approach, as well as applying economic and industry efficiency analysis of a West German rationale, this comment concludes with a proposal for a viable alternative to current intellectual property protection.

II. BACKGROUND

A. THE FALLACY OF A SOFTWARE-HARDWARE DISTINCTION

"The computer is no better than its program."[17]

Historically, the computer industry divides its technological endeavors into three types: (1) hardware: the physical device itself, collection of transistors, and integrated circuits (i.e., semiconductors or "microchips");

12. Thomas P. Burke, *Software Patent Protection: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1115-16 (1994). Due to inefficient and ineffective intellectual property protection, the software industry suffered a $9.7 billion dollar loss in 1992, and a $7.4 billion dollar loss in 1993. *Id.* at 1115. The article attributes the reduction of losses in 1993 to the increased protection provided by the newly instituted "software patent." *Id.*

13. 35 U.S.C. §§ 102-12 (1988). "Prior Art" is the material that the patent office compares with the application material to ascertain whether the material applying for a patent is a useful "point of novelty." *Id.*

14. Luettgen, *supra* note 8, at 240-41 (noting that copyright protects the "form of computer software," but cannot protect the "mechanical and utilitarian aspects" of software).

15. Luettgen, *supra* note 8, at 262-63 (showing how copyright law promotes disclosure because copyright traditionally protect communicative works).

16. Ballon & Rafter, *supra* note 3, at 105 (explaining the inconsistent application of patents by the Patent and Trademark Office).

17. JOHN BARTLETT, BARTLETT'S FAMILIAR QUOTATIONS 876 (15th ed. 1980) (quoting Elting Elmore Morison, *Men, Machines and Modern Times*).

(2) software: the code that ultimately resides in the microchips, telling the hardware what to do; and (3) algorithms: the purely abstract routines for accomplishing the processing goals of the system.[18] Both software and hardware can express any algorithm, making software and hardware functionally equivalent.[19] Despite this fact, courts, in most cases, consistently apply copyright law to software and patent law to hardware.[20] This continuing denial of a hardware—software equivalence has lead to judicial inconsistencies and "scientifically untenable decisions."[21] Thus, the similarity of software and hardware requires a similar means of protection.[22]

---

18. Note, *Computer Intellectual Property and Conceptual Severance*, 103 HARV. L. REV. 1046, 1047 (1990).

19. *Id.* For example, software can express an algorithm in a programming language, while hardware can express the same algorithm in an integrated circuit. *Id.* Upon activation, the same algorithm in software or hardware generates the same end intended by the programming language. *Id.*

20. Ronald S. Laurie & Jorge Contreras, *Application to the Doctrine of Equivalents to Software-Based Patents*, 292 PLI/PAT 689, 710-11 (1990). As early as 1964, courts blurred the distinction between hardware and software by noting that hardware and software are merely different means of achieving the same result. *Id.* at 710. *See also* Bullard Co. v. General Elec. Co., 348 F.2d 985, 992 (W.D. Va. 1964) (holding that although both electronic machine controllers and patented mechanical machine controllers did not function the same way, both machines achieved the same results); Hughes Aircraft Co. v. United States 717 F.2d 1351 (Fed. Cir. 1983) (holding that although hardware and software differed in efficiency and in their respective methods of achieving a function, nevertheless, the functions were equivalent); Pennwalt Corp. v. Durand-Wayland, Inc., 225 U.S.P.Q. 558 (N.D. Ga. 1984), *aff'd en banc*, 833 F.2d 931 (Fed. Cir. 1987), *cert. denied*, 485 U.S. 961 (1988).

21. James R. Goodman, et al., *Toward a Fact-Based Standard for Determining Whether Programmed Computers are Patentable Subject Matter: The Scientific Wisdom of Alappat and Ignorance of Trovato*, 77 J. PAT. [& TRADEMARK] OFF. SOC'Y 353, 361 (1995). The article argues that in *In re* Alappat, 33 F.3d 1526, 1543 (Fed. Cir. 1994), the court accurately observed that the judicially created exception of algorithms to § 101 of the Patent Act is trumped by the congressional intent of § 101, namely that § 101 should "cover anything under the sun." *Id.* The article notes that of the four recent decisions in the Federal Circuit, *In re* Alappat, 33 F.3d 1526; *In re* Warmerdam 33 F.3d 1354 (Fed. Cir. 1994); *In re* Lowry 32 F.3d 1579 (Fed. Cir. 1994); and *In re* Trovato 42 F.3d 1376 (Fed. Cir. 1994), only *In re* Trovato held a computer program unpatentable. *Id.* Consequently, the hardware-software distinction created inconsistency in four Federal Circuit cases in the same year. *Id.*

22. A. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION 11 (2d ed. 1984). The author states the software-hardware similarity as follows: "hardware and software are logically equivalent . . . any operation performed by software can also be built directly into hardware and any instruction executed by hardware can be simulated in software . . . . [T]he decision to put certain functions in hardware and others in software is based on such factors as cost, speed, reliability, and frequency of expected changes." *Id.* at 11-12.

B. Copyright Law and Computers

1. *Copyright In General*

Congress, through the National Committee on New Technological Uses of Copyrighted Works ("CONTU"),[23] declared that computer programs, as intellectual property, are entitled to copyright protection.[24] Traditionally, copyrights protect original works that are fixed in a tangible medium of expression where such medium is "perceived, reproduced, or otherwise communicated."[25] However, the Copyright Act specifically limits this by denying protection for "ideas, procedures, processes, or methods of operation."[26] For a corporate entity, a copyright grants protection to the material in question for a period of seventy-five years from the date of publication, or one-hundred years from the date of creation, whichever expires first.[27] Unlike patent protection,[28] the protection of copyrighted material begins as soon as the work becomes "fixed."[29] In addition, the issue of whether an infringement occurs merely because two works are similar, or identical, is never determined by copyright law.[30] This issue turns on whether the works were independently created.[31] Therefore, to establish a claim of copyright infringement, the plaintiff must prove: (1) ownership of a valid copyright, and (2) copying of constituent elements of the work that are original.[32]

---

23. Arthur Miller, *Copyright Protection For Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 Harv. L. Rev. 977 (1993). In 1976, CONTU considered computer programs as "literary works" for the purpose of intellectual property protection. *See* H. R. Rep. No. 1476, 94th Cong., 2d Sess. 116 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5731. This instruction by CONTU was codified under 17 U.S.C. § 101. *Id.*

24. 17 U.S.C. § 101 (1988 & Supp. 1990). A computer program is "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result," thereby making computer programs eligible for protection under a copyright theory. *Id.*

25. 17 U.S.C. § 102(a) (1988 & Supp. 1990).

26. 17 U.S.C. § 30(b) (1988 & Supp. 1990) (stating the limits of copyright protection).

27. 17 U.S.C. § 30(a), (b) (1988 & Supp. 1990).

28. 35 U.S.C. § 102 (1988 & Supp. 1990). Pursuant to this statute, the protection for patented material does not begin until the patent is granted. *Id.*

29. 17 U.S.C. § 102(a) ("fixed in any tangible medium of expression"). *See also* Miller, *supra* note 23, at 987. Miller contends that computer programs are sufficiently "fixed" for copyright law to protect them. Miller, *supra* note 23, at 987. Before CONTU, debate took place over whether a computer program is sufficiently "fixed" for the purposes of the Copyright Act. Miller, *supra* note 23, at 988. CONTU decided that due to the amount of time that a program is inside the memory, and the hypothetical possibility that a computer could remain running ad infinitum, a computer program is "fixed," and therefore, copyrightable. Miller *supra* note 23, at 988.

30. Feist Publications Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 361 (1991).

31. *Id.*

32. *Id.* The *Feist* court noted that in most computer software cases, the first prong—proving ownership of the copyright—is basically a technicality. *Id.* at 362.

2. *Copyright for Computers*

a. Literal Elements and the Fair Use Exception

In deciding a software copyright issue, the courts must understand and determine the extent of copyright protection for the "elements" of a computer program.[33] These elements are subdivided in two parts, literal and nonliteral. "Literal" elements are the source code and object code of the program,[34] while "nonliteral" elements are the structure, sequence, organization, and computer-user interface of the program.[35] When protecting the literal elements of software, most defendants assert that "re-

33. 17 U.S.C. § 101 (1988). Congress amended the Copyright Act in 1980 to specifically include software, making only the literal elements software protectable under copyright. *Id.* Since 1980, many courts have upheld the notion of protectability for both the source code and the object code of software. *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1246-47 (3d Cir. 1983) (holding that both object code and source code are protectable); Midway Mfg. Co. v. Strahon, 564 F. Supp. 741, 749-52 (N.D. Ill. 1983) (holding that even when the object code or source code is embodied in a computer chip, the respective codes are protectable); Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 855 (2d Cir. 1982) (holding that source code is protectable); Tandy Corp. v. Personal Micro Computers, Inc., 524 F. Supp. 171, 173 (N.D. Cal. 1981) (holding for the protectability of both types of codes in computer chips).

34. Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1341 (5th Cir. 1994). The source code is the programming language that is readable by human programmers. *Id.* The source code is then translated through a process, known as compilation or assembly, into the "object code." *Id.* The object code is the binary expression, readable by the computer, that controls the computer hardware. *Id.* A skilled programmer can read and understand small sections of the object code, but a programmer cannot develop a working understanding of a program by viewing only its object code. Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1542 (11th Cir. 1996).

35. TANENBAUM, *supra* note 22, at 12. The following are explanations of key terms in computer copyright technology: first, "computer hardware" consists of the machine itself, while computer software is the set of instructions, written by programmers that tell the hardware to perform certain tasks. TANENBAUM, *supra* note 22, at 12. These instructions are in the form of mathematical computations known as "algorithm." TANENBAUM, *supra* note 22, at 12. Software or hardware can express any algorithm. TANENBAUM, *supra* note 22, at 12.

Next, software is usually divided into two categories: operating systems and application systems. TANENBAUM, *supra* note 22, at 13. Operating systems control the internal operations of the computer by transferring data from one point to another, while application systems tell the operating systems to instruct the computer to perform a certain task like word processing. TANENBAUM, *supra* note 22, at 13.

Finally, the hardware, operating system, application software and user communicate through "interfaces." Timothy S. Teter, Note, *Merger and the Machines: An Analysis for the Pro-Compatibility Trend in Computer Software Copyright Cases*, 45 STAN. L. REV. 1061, 1063 (1993). The system communicates with the user through a "user interface" which could consist of images on a monitor, as well as the keyboard or mouse. *Id.* For the entire system to function, the components must be "compatible," i.e., communication must flow uninterrupted throughout the system. *Id.*

verse engineering"[36] of a computer program constitutes a "fair use"[37] under the Copyright Act.[38] The fair use defense, also known as the fair use exception, is applicable when a programmer has no alternative but to copy the program verbatim in order to gain an understanding of the uncopyrightable ideas and functional aspects of a work.[39]

b. Nonliteral Elements and the A-F-C Test Variations

It was not until recently that courts began to focus on the possibility of copyright protection for nonliteral elements of a program.[40] Courts have struggled with applying copyright law to the nonliteral elements of a program. More specifically, they are not able to ascertain whether the nonliteral elements contain protectable material pursuant to the Copyright Act, or merely ideas and functions that are not protected under the Act.[41] Many courts are attempting to separate the nonprotectable functional elements from the protectable expressive elements, via the "abstraction-filtration-comparison" test ("a-f-c" test).[42] This test is applied

36. Mark Aaron Paley, *A Model Software Petite Patent Act*, 12 SANTA CLARA COMPUTER & HIGH TECH. L.J. 301, 343 (1996). Generally, "reverse engineering" refers to a variety of activities used to reveal the design of computer software. *Id.* For example, "Black Box" reverse engineering analyzes the input and output of a program without viewing the program's internal design. *Id.* This kind of reverse engineering attempts to make a program compatible with another. *Id.* More intrusive reverse engineering "disassembles" or "decompiles" software via special software used to translate machine object code into human-readable source code. *Id.* This decompiled code can reveal the design and engineering of the computer program. *Id.*

37. 17 U.S.C. § 107 (1988). Section 107 of the Copyright Act provides an affirmative defense to allegations of copyright infringement by asserting that the copying of the product in question was done as a "fair use." *Id.* Specifically, the statute suggests balancing four factors in order to assert the fair use defense: (1) the purpose and character of the use, taking into consideration whether the use is commercial or nonprofit in nature; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the fair use upon the potential market for or value of the copyrighted work. *Id.* An example of fair use of copyrighted material is material photocopied for educational purposes. *Id.*

38. Paley, *supra* note 36, at 344.

39. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527-28 (9th Cir. 1993). *See also* Atari Games Corp. v. Nintendo of Am., Inc. 975 F.2d 832, 844-45 (Fed. Cir. 1992) (demonstrating two examples of the "fair use" defense asserted in copyright infringement proceedings of reverse engineering of computer programs).

40. Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc. 797 F.2d 1222, 1233-35 (3d. Cir. 1986), cert. denied, 479 U.S. 1031 (1987). This was the first case to define the scope of copyright protection for nonliteral elements. *Id.*

41. *Altai*, 982 F.2d at 712. The confusion of the current state of copyright protection for computer software mentioned in *Altai* is mostly due to the difficulty in separating the protectable expressive material from the nonprotectable functional material. *Id.*

42. Gates Rubber Co. v. Bando Chem. Indus., Ltd. 9 F.3d 823, 834 (10th Cir. 1993).This court stated the a-f-c test as follows:

by the majority of the courts.[43] However, Judge Keaton in *Lotus Development Corporation v. Paperback Software International (Lotus I)*,[44] noted that lawmakers cannot deceptively rely on a formula such as the a-f-c test in its entirety.[45] Instead, Keaton in *Lotus I* favors a strict case-by-case analysis with regards to computer software cases.[46]

In 1981, the Supreme Court in *Diamond v. Diehr*[47] opened the door for the potential patentability of scientific principles or abstract ideas based upon mathematical computations embodied in computer software.[48] Subsequent to this Supreme Court decision, Congress finally recognized the need for increased and specialized protection of semiconductor chip products.[49] However, this same special level of protection

---

> First, in order to provide a framework for analysis, we conclude that a court should dissect the program according to its varying levels of generality as provided in the abstractions test. Second, poised with this framework, the court should examine each level of abstraction in order to filter out those elements of the program which are unprotectable. Filtration should eliminate for comparison the unprotectable elements of ideas, processes, facts, public domain information . . . and other unprotectable elements suggested . . . . [T]hird, the court should then compare the remaining elements with the allegedly infringing program to determine whether the defendants have misappropriated substantial elements of the plaintiff's program.

Id.

43. *See generally* Mitek Holdings, Inc. v. ARCE Eng'g Co., 864 F. Supp. 1568 (S.D. Fla. 1994); Altai, 982 F.2d 693; Gates Rubber, 9 F.3d 823; Engineering Dynamics, 26 F.3d 1335. The Supreme Court recently upheld, in a 4-4 decision, with Justice Stevens not participating, Judge Keaton's version of the *Altai* a-f-c test. Lotus Dev. Corp. v. Borland Int'l, Inc., 116 S. Ct. 804 (1996). The standard of the a-f-c test, as upheld by the Supreme Court, states:

> First, in determining copyrightability, one must focus upon alternatives from any source, to some conception of definition of the "idea, "system," "process," "procedure," or "method"—for the purpose of distinguishing the idea, system, purpose, process, procedure, or method, from the expression.
>
> Second, one must focus upon whether an alleged expression of the idea, system, process, procedure, purpose, or method is limited to elements essential to the expression of that idea, system, process, etc., or instead includes identifiable elements of expression not essential to every expression of that idea, system, process, etc.
>
> Third, having identified the elements not essential to every expression of the idea, system, process, etc., one must then focus on whether those expressive elements, taken together, are a substantial part of the work.

Lotus II, 788 F. Supp. at 90.

44. 740 F. Supp. 37 (D. Mass. 1990).

45. *Id.* at 52.

46. *Id.*

47. 450 U. S. 175 (1981).

48. *Id.* at 184

49. 17 U.S.C. §§ 901-06 (1984). In 1984, the Semi-Conductor Chip Protection Act was created to deal especially with the nuances created by computer technology. *Id.* The act defines exactly what and how electronic circuitry functions are protected by copyright, as well as the ownership and licensing of copyrights of electronic property, and the limitations of the copyright through reverse engineering. *Id. See supra* note 36 and accompanying text (defining "reverse engineering").

was not afforded software. Thus, the interchangeable funtionality of software and hardware leads to the conclusion that software needs the same patent protection as provided hardware.[50]

C. Patent Law and Computers

1. *Patent Law in General*

Generally, patent statutes are intended to protect creative works for utilitarian objects and processes.[51] Patents are granted to whomever invents or discovers any process, machine, manufacture, matter, composition, or any improvement thereof that is a novel,[52] useful,[53] and non-obvious[54] invention.[55] Patent law contains the implicit notion that the production of some "prior art" is needed to demonstrate the desirability and, therefore, non-obviousness of the contribution.[56] Also, a patent for useful combinations (commonly known as "means-plus-function claims") is obtainable.[57]

Unlike copyrights, patents are granted for shorter periods of time, and the scope of patent rights are relatively broad.[58] Moreover, to facilitate innovation and increase the flow of information, patent statutes re-

---

Congress does not provide the same increased level of protection for the software industry that Congress provides for the semiconductor chip industry. The difficulty of differentiating between hardware and software, coupled with the extreme growth of the software industry, points to Congress taking the same measures for protection of software as it did with semiconductors. *See also* Kidwell, *supra* note 7 (noting the difference of protection in software and semiconductors, the similarity in the two industries, and the need for increased software protection).

50. *Hearing on The Intellectual Property Antitrust Protection Act Before the House Judiciary Committee*, H.R. 2674 (Tues., May 14, 1996) (congressional testimony of Jacob Frank, industry attorney). Frank notes the hardware/software similarity is reflected in the computer industry by stating that fourteen companies are developing operating system software for their respective hardware. *Id.*

51. 35 U.S.C. §§ 1-376 (1988).

52. 35 U.S.C. § 101 (1988). "Novel" means that the invention must be "new . . . subject to the conditions of this title." *Id.*

53. 35 U.S.C. § 101 (1988) (prohibiting inventions which are frivolous or injurious to the well-being of society, such as mischievous or immoral inventions).

54. 35 U.S.C. § 103 (1988). "[O]bviousness is a legal determination . . . to a person having ordinary skill in the art at the time the invention was made." *Id.*

55. 35 U.S.C. § 101 (1988) (explaining the need for patentable material to contain all three requirements of novelty, usefulness, and non-obviousness).

56. 35 U.S.C. §§ 102-12 (1988) (defining "prior art.")

57. 35 U. S. C. § 112 ¶ 6 (1988). A "means-plus-function" claim is: "An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or act in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof." *Id.*

58. *Compare* 35 U.S.C. §§ 102-03 (1988) *with* 17 U.S.C. § 102(b) (1988 & Supp. 1990) (comparing the scope and length of protection for patents and copyrights, respectively).

quire that the inventor describe his or her invention, proving that the invention is distinguishable and the preferred version or "best mode" of the invention as of the filing date.[59] Accordingly, the inspection of prior art and proof of best mode and usefulness requires more effort and is, therefore, more expensive than a copyright application.

2. *The Patentability of Computer Programs*

Over the past three years, courts have frequently accepted and implemented software patents.[60] However, these cases are not without issue. In attempting to match patent law to software, lawmakers address the following two issues: (1) whether the software, hardware, or algorithm is patentable matter; and (2) whether patent law requirements of written disclosure to describe, enable, and provide the best mode for practicing an invention are consistent with the software industry.[61] For example, the courts deny protection for reciting an algorithm, yet provide protection for an algorithm-based invention as long as the invention could provide novelty in a means-plus-function format.[62]

In attempting to comply with section 112 of the Patent Act, the "enablement" requirement mandates that the party seeking a patent disclose the invention to a point where others have the power to replicate the invention.[63] The courts, however, are undecided as to whether a

---

59. 35 U.S.C. § 112 (1988). "Best mode" analysis has two requirements. *Id.* The first inquiry focuses on, at time of filing the application, whether the inventor considered that the mode used in the invention is considered better than any other. *Id.* This is a subjective determination, focusing on the applicant's state of mind. *In re* Hayes Microcomputer Prods., 982 F.2d 1527 (Fed. Cir. 1992). If the inventor did have a best mode, then the Patent and Trademark Office determines if the disclosure of the invention is adequate so that another can practice the invention. *Id.* This is an objective determination. *Id.* The inventor cannot conceal a better method for performing or manufacturing the invention. *Id.*

60. *See* Goodman, *supra* note 21, at 354 (questioning the rationale of the *Trovato* decision, which is one of the few decisions to deny patentability to a computer program).

61. 35 U.S.C. § 112 ¶ 1 (1988) (describing the disclosure requirements mandated by the Patent Act).

62. Gustavo Siller, Jr. & Jonathan E. Retsky, *Patent and Trade Secret Protection of Computer Technology*, 6 SOFTWARE L.J. 239, 246-54 (1993) (providing an excellent summation and judicial history of the struggle to define the new software patent). *See also supra* note 57 (defining a "means-plus-function claim").

63. 35 U.S.C. § 112 (1988). *See also* Burke, *supra* note 12, at 1125. A software patent is often compared to a drug patent. Burke, *supra* note 12, at 1125. Thus, some courts use the enablement guidelines for drug patents and apply these to software. Burke, *supra* note 12, at 1125. These factors summarized by the Federal Circuit Court are: (1) the quantity of experimentation necessary, (2) the amount of direction or guidance presented, (3) the presence or absence of working examples, (4) the nature of the invention, (5) the state of the prior art, (6) the relative skill of those in the art, (7) the predictability or unpredictability of the art, and (8) the breadth of the claims. *In re* Wands, 858 F.2d 731, 737 (Fed. Cir. 1988).

programmer must disclose the source code[64] in the patent application.[65] If a source code is not required, flow-charts and other information must provide adequate direction for a programmer of average skill to implement the program without "undue experimentation."[66]

Unlike the enablement requirement, the best mode requirement involves a subjective inquiry that requires evidence that the inventor knowingly concealed the "best mode" of the invention.[67] Under best mode, the court in *In re Sherwood*[68] decided that in most cases, the actual disclosure of the entire program is required.[69] The court also noted that lack of disclosure, resulting in concealment, causes the patent to be denied.[70]

The Patent and Trademark Office ("PTO") cannot accept or deny patent applications for computer programs with the same efficiency and confidence as other types of inventions due to the absence of prior art with which to compare these programs.[71] The lack of efficiency and confidence delays innovation and the free flow of information, as well as increases the social and actual costs of obtaining a software patent.[72] The lack of software patent prior art brings the PTO to a paradox. This paradox is in order to issue patents efficiently, the PTO must possess adequate prior art; yet, in order to obtain an adequate quantity of prior art, the office must issue patents. The following section sets forth a solution to this paradox.

III. ANALYSIS

For computer software, intellectual property protection raises questions involving an infusion of answers from science, technology, econom-

64. The instructions a programmer writes to control a computer's future operation—to make the program compatible with the respective system.

65. Lawrence Kass, *Computer Software Patentability and the Role of Means-Plus-Function Format in Computer Software Claims*, 15 PACE L. REV. 787, 792 (1995) (arguing that Congress should have determined whether applicants should disclose the source code, object code, flowcharts, or a combination thereof).

66. *In re* Vaeck, 947 F.2d 488 (Fed. Cir. 1991). Enablement empowers others to replicate the patented invention. *Id.* at 492. Although not specifically stated in the statute, enablement requires that "those in the art can make and use the invention without 'undue experimentation.'" *Id.* Some experimentation is required, as long as the amount of experimentation is not "fatal." *Id.*

67. *See supra* note 59 (defining "best mode").

68. 613 F.2d 809 (C.C.P.A. 1980), *cert. denied*, 450 U.S. 994 (1981).

69. *Id.* at 815.

70. *Id.* The court also provided the penalties for concealment and nondisclosure of the entire computer program in applying for a software patent. *Id.*

71. Paul A. Mendonsa, *Patent Protection for Multimedia Products*, 467 PLI/PAT 235, 249 (1997) (stating that the PTO examiners cannot show the statutory requirements of obviousness and novelty due to the lack of prior art).

72. *See infra* note 128 (discussing transaction cost analysis and the Coase theorem).

ics, law, and politics.[73] Today, the law provides a "patchwork of legal protection" applying patent and copyright protection to software.[74] This application reveals the inability of patent or copyright law to adequately protect the actual and abstract portions of the intellectual property of computer software. Nonetheless, only the law possesses the true authority to protect the inventions of computer technology that, by their very nature, are expensive to develop but inexpensive to copy.[75] In light of the potential eradication of the software industry's intellectual property value, as well as the industry's eventual competitive market failure, the software industry requires new intellectual property protection as its saving grace.[76]

A. Pros and Cons of Copyrights for Computer Software

1. *The Positive Aspects of Copyrights for Software*

CONTU equates computer programs to "literary works"[77] for classification and protection purposes.[78] In some instances, this classification of "literary work" works to the advantage of software manufacturers. First, a copyright allows a programmer, to a certain extent, to protect the "creativity, originality, and insight" that was involved in the process of conceptualizing a computer program.[79] To this end, copyright law allows the programmer the protection of individuality within the confines of binary language and nonliteral elements, just as a poet conveys a complex

73. Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 Colum. L. Rev. 2644 (1994).

74. Note, *supra* note 18, at 1049.

75. *See* Kidwell, *supra* note 7, at 533. Kidwell observes that both the software industry and the semi-conductor chip industry have the unique combination of high development costs and low copying costs. Kidwell, *supra* note 7, at 588. By analogy, Kidwell argues that Congress should protect software as it protected semi-conductors in the Semi-Conductor Chip Protection Act of 1984. Kidwell, *supra* note 7, at 588. *See also* 17 U.S.C. §§ 901-06 (1984).

76. *See Manifesto*, *supra* note 6, at 2430 (highlighting the current market-destructive behavior of the current intellectual property system through the misappropriation of program behavior within the intellectual property system).

77. Miller, *supra* note 23, at 988-89.

78. 17 U.S.C. § 101 (1988). The post-CONTU version of the Copyright Act states that computer programs are literary works: "Works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phono records, film, tapes, disks, or cards, in which they are embodied." *Id.*

79. Lotus Dev. Corp. v. Borland Int'l, Inc. (Lotus II), 788 F. Supp. 78, 96 (D. Mass. 1992). In this case, the court alludes to the possibility of patent protection encompassing the ideas included in software. *Id.* Notwithstanding, Judge Keaton still advocates the use of copyrights for most computer programs, while realizing the uselessness of bright-line rules and the difficulty of foreseeing technology levels and according levels of intellectual property protection. *Id.*

message within the confines of the medium of poetry.[80] Second, the limited scope and lower standards of the copyright makes the copyright much easier to obtain, compared to other forms of intellectual property protection.[81] The copyright's ease of obtainability avails itself perfectly to the incremental innovation of the computer industry.[82] Third, comparatively speaking, computer companies obtain and maintain copyright protection for far less cost than other intellectual property protection.[83] And finally, protection of the expressive work begins automatically with the creation of the work.[84] Through these means, the copyright unquestionably provides an easy, quick, and economical way for software companies to protect expressive[85] elements of software.

2. *The Negative Aspects of Copyrights for Software*

According to *Computer Associates International v. Altai*,[86] copyright law aims to establish a "delicate equilibrium [between] afford[ing] protection to authors as an incentive to create, and . . .appropriately limit[ing] the extent of . . . protection to avoid monopolistic stagnation."[87] Unfortunately, the current application of copyright law to computer

80. *See* Miller, *supra* note 23, at 984. Extending the "literary work" metaphor, Miller states: "Just as no two novelists independently would compose the same detailed plot of the downfall of a tragic hero . . . no two programmers independently would design a program that enabled the computer to solve highly intricate problems with the same structural details, let alone the precisely the same set of instructions." *Id.*

81. Pamela Stern, *The Bundle of Rights Suited to New Technology*, 47 U. PITT. L. REV. 1229, 1247 (1986). Because the patent protects more of the intellectual property, namely the idea and functionality of the product, the patent is more difficult to obtain than a copyright with regards to time, effort, proof of novelty, and non-obviousness. *Id.*

82. *See Manifesto*, *supra* note 6, at 2346. The *Manifesto* observes that Congress realized the incremental nature of the computer industry by passing the Semi-Conductor Chip Act of 1984. *Manifesto*, *supra* note 6, at 2346. The *Manifesto* then argues that Congress should provide the same sort of protection for computer software. *Manifesto*, *supra* note 6, at 2346.

83. Himanshu S. Amin, *The Lack of Protection Afforded Software Under the Current Intellectual Property Laws*, 43 CLEV. ST. L. REV. 19, 22 (1995). The approximate cost of obtaining a software patent exceeds $10,000. *Id.* The excessive cost is associated with the effort of preparing the application for a patent, filing the application, and the lengthy examining of the application by the Patent and Trademark Office. *Id.*

84. *Id.* at 23.

85. Recall, the copyright only protects the "expressive" portion of the work, leaving the remaining "idea, procedure, process, system, method of operation, concept, principle, or discovery" unprotected. *See* 35 U.S.C. §§ 102-103 (1988) *and* 17 U.S.C. § 102 (1988 & Supp. 1990) (comparing the scope of copyright and patent protection).

86. 982 F.2d 693 (1992).

87. *Lotus II*, 788 F. Supp. at 96. The court states that all courts must keep the maintenance of this "delicate equilibrium" in mind when applying the federal copyright act to software. *Id.* The court further notes that as of this opinion other courts "grappled" unsuccessfully when applying the Copyright Act to the non-literal aspects of computer software. *Id.*

software creates a "disequilibrium" of underprotection, thereby establishing a lack of incentive to create.[88] Although the courts are providing an increasing amount of protection for the copying of software through protection of the program's non-literal[89] structures,[90] a consensus of legal scholars contend that copyright law cannot protect the utilitarian aspect of software from which computer software derives its value.[91]

Protecting nonliteral elements through the a-f-c test conflicts with the fair use of reverse engineering. Such is the case, because courts do not have a definitive answer as to what nonliteral elements are protected, or whether these elements can be copied under reverse engineering.[92] Nevertheless, this issue of reverse engineering and protection of the nonliteral elements remains unsettled. Consequently, courts strug-

88. Slind-Flor, *supra* note 1, at A2. The computer industry is experiencing excellent growth and profits; however, the potential profits that software companies could reap with increased property protection makes the current profits seem minute in comparison. Slind-Flor, *supra* note 1, at A2.

89. *Altai*, 982 F.2d at 702. Because the Copyright Act, 17 U.S.C. §101, protects nonliteral portions of literary works, and Courts hold that Congress intended computer programs to function as "literary work" for the purpose of the Copyright Act, therefore, the Copyright Act protects the non-literal portions of computer programs. *Id.*

The courts decided that the following are not protectable under copyright law: *Lotus Dev. Corp.*, 788 F. Supp. at 89 (denying protection for menu command hierarchies); *Apple Computer*, 799 F. Supp. at 1034 (denying protection for icons); *Apple Computer*, 799 F. Supp. 1006, 1034 (N.D. Cal. 1992) (denying protection for use of windows to displays and multiple images for user interaction); *Apple Computer*, 799 F. Supp. at 1035 (denying protection for use of menus to store information or computer functions); *Apple Computer*, 799 F. Supp. at 1035 (denying protection for opening and closing of objects as a means to store, retrieve and transfer information); *Engineering Dynamics*, 26 F.3d at 1346 (denying protection for input/output formulas); *Altai*, 982 F.2d at 715 (denying protection for the underlying code of screen displays).

90. *See, e.g.*, Phoenix Controls, Inc. v. Phoenix Control Sys., 886 F.2d 1173, 1175 (9th Cir. 1989); Digital Communications Assocs., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455-56 (N.D. Ga. 1987); Q-Co Industries, Inc. v. Hoffman, 625 F. Supp. 608, 615 (S.D.N.Y. 1985); SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 829-30 (M.D. Tenn. 1985).

91. *See generally* David Bender, *Protection of Computer Programs: The Copyright/Trade Secret Interface*, 47 U. PITT. L. REV. (1986) (comparing copyright to trade secret law with regards to the protection provided under each); Menell, *supra* note 62, at 1054 (arguing for application of copyright idea/expression merger doctrine to allow diffusion of scientific ideas); Office of Technology Assessment, United States Congress, *Finding a Balance: Computer Software, Intellectual Property and the Challenge of Technology Change* 5 (1992) (noting the difficult questions in copyright case law concerns whether functionality, and not just the code, is protected); *Altai*, 982 F.2d at 704-12 ("[T]he utilitarian nature of a computer program . . . complicates the task of distilling its idea from its expression"); *Lotus II*, 788 F. Supp. at 91-92 (admitting the struggle to find copyright protection for the functionality aspects of software).

92. Andre R. Jaglom, *Current Developments in Copyright Protection of Computer Software*, CA63 ALI-ABA 603, 606 (1996). By protecting the nonliteral elements—the "structure, sequence, and organization" of the program, a would-be reverse engineer cannot

gle in their attempt to separate the protected expressions from the non-protected ideas,[93] resulting in continued inconsistencies[94] with respect to which test adequately separates the copyright-protectable expression from the non-protectable idea.[95]

B. Pros and Cons of Patents for Computer Programs

1. *The Positive Aspects of Patents for Software*

With increased protection, the patent system promotes innovation and the dissemination of technologies.[96] The Patent Act requires a writ-

---

legally copy the program to reveal either the program's "structure, sequence and organization," or the program's ideas and functionality, not protectable under copyright. *Id.*

93. *See supra* notes 42-44 (describing the a-f-c test). In utilizing the above mentioned a-f-c test in separating protectable elements from the nonprotectable, and literal elements from nonliteral, the courts assume that they can separate the literal and nonliteral elements. *See supra* notes 42-44. Specifically, the courts assume that their "filter" is fine enough to catch all of the protectable nonliteral elements. *See supra* notes 42-44 (describing the a-f-c test).

94. *See, e.g., Whelan*, 797 F.2d at 1236. The court used the following approach to separate idea and expression:

> [T]he purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression or idea . . . . [W]here there are various means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.

Id.

The *Altai* court notes a flaw in the *Whelan* court's reasoning by noting that ". . . [I]t (the *Whelan* court) assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression." *Altai*, 982 F.2d at 705. Accordingly, the *Altai* court utilizes the classic abstraction-filtration-comparison method in an attempt to separate ideas from the expressions. *Id.* The court quotes the CONTU report when it states,

> Copyrighted language may be copyrighted without infringing when there is but a limited number of ways to express a given idea . . . . [I]n the computer context, this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement.

See Miller, *supra* note 23, at 121.

The *Engineering Dynamics* court, on the other hand, examines many deviations of the abstraction-filtration-comparison test and ultimately comes to the conclusion that the "sweat of the brow" test and the Zack Meyer Originality test are applicable in addition to the abstraction-filtration-comparison test followed by the *Altai* court. Engineering Dynamics, 26 F.3d at 1341; *see also* Feist, 499 U.S. at 359; Donald v. Zack Meyer's T.V. Sales and Service, 426 F.2d 1027 (5th Cir. 1970).

95. Gary A. David, *Lotus v. Borland: A Step Forward on ideas, A Step Back on Expression*, 450 PLI/Pat 339, 369 (1996). Very few cases, if any, adequately draw the line between idea and expression utilizing any method or version of the a-f-c test. *Id.* The Supreme Court's non-decision in the *Lotus I* case did nothing for the resolution of the idea-expression paradox, leaving more unanswered questions than it settled. *Id.*

96. *Id.* The irony of patent law in this particular instance is that a patent monopoly actually increases competition. *Id.* With the reluctance of companies to send licensing

ten description of the potentially patentable invention.[97] This written disclosure requires the best mode[98] of the invention, while enabling[99] others to recreate the invention. The written disclosure provision of the Patent Act forces efficiency by requiring the "best mode" for practicing the invention, while simultaneously stimulating further advances by enabling others to practice the invention.[100] In return for this disclosure, the patentee receives the right to exclude others from profiting from the invention. Thus, the patent-granted monopoly of the invention creates a figurative bargained-for-exchange: in exchange for an informative disclosure of an innovation, similar to the information gained through reverse engineering, the patent holder receives a monopoly and a right of excludability.[101]

2. *The Negative Aspects of Patents for Software*

Traditionally, patents are unable to protect text or "printed matter" and mental "processes," thus, on their face patents appear incompatible with computer programs.[102] However, some courts apply a patent law rationale to software, and as a result, realize the functionality aspect of software.[103] The valuable elements of a computer program, the behav-

---

checks to their respective competitors, competitors must "put forth their mightiest efforts to produce a product as good, yet different from the patent-holder's." *Id.* at 1130. Antitrust laws protect consumers and foster free competition; the patent laws must guarantee that new entrants into the market are adequately protected to maintain efficient competition. *Id.* at 1130-31.

97. 35 U.S.C. § 112 ¶ 6 (1988).

98. 35 U.S.C. § 112 ¶ 1 (1988). *See also supra* note 59 (describing "best mode" requirements).

99. *Hayes,* 982 F.2d at 1533-37. "Enablement" requires that a person of ordinary skill in the respective field is able to recreate the invention via the written description. *Id.*

100. *See* Burke, *supra* note 12, at 1139 (noting that the best mode requirement requires the applicant to "play fair and square" with the patent system).

101. Lawrence D. Graham & Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure,* 22 RUTGERS COMPUTER & TECH. L.J. 61, 96 (1996). The article argues for the economically and socially efficient free exchange of information that reverse engineering provides to the software market. *Id.* The disclosure requirements of best mode and enablement in applying for a patent provides the same effect of free-flow of information, while adequately compensating the disclosure. *Id.* at 97. Therefore, disclosure in patent is analogous to reverse engineering in copyright, except that in copyright the use of the information gained through reverse engineering is not compensated by monopoly excludability as in patent law. *Id.* at 98-99.

102. *See, e.g.,* United States Patent and Trademark Office & United States Copyright Office, *Patent-Copyright Laws Overlap,* Study 46 (1991). Printed matter is traditionally the basis from which software is denied patent protection. *See, e.g., In re* Abrams, 188 F.2d at 168 (holding that "calculating," "measuring," "observing," and "comparing" data elements are unpatentable).

103. *Alappat,* 33 F.3d at 1544. Before the landmark *Alappat* holding, the legislature did not consider mathematical algorithms patentable under 35 U.S.C. § 101 of the Patent Act.

ior, does not lend itself to patent protection: patents typically protect particular methods of achieving results, not the results themselves.[104] In other words, a patent holder with one method of generating certain results could not prevent the use of another method to obtain the same results or behavior.[105] Nonetheless, the major problem involved in applying patent protection to software is that the innovations in the computer industry, whether hardware, software, or semiconductors,[106] are incremental in nature.[107]

The system of patent law is designed to protect the innovations of a substantial contribution to an industry or invention.[108] Under the current system, those companies that need the increased protection of patents either cannot afford the time and effort of obtaining a patent, or did not make the requisite leap in innovation to warrant the issuing of a patent.[109] The lengthy and costly application process does not generously lend itself to the small software manufacturer.[110] Therefore, the small firm that the software patent was designed to protect cannot even apply for a patent due the costly and inefficient application process. Under the current system, the issuing of patents in an attempt to increase the needed protection of computer software is like bringing a gar-

---

Now, when the courts consider software as an actual process controlling hardware, then the software is patentable. *Id.* The software invention must be a part of the "specific machine to produce a useful, concrete and tangible result," as well as meet the statutory novelty and non-obvious criteria. *Id.*

As early as 1976, the Supreme Court held that if the software process controlled a physical operation or changed a state of matter, then the process is patentable. *See generally* Damn v. Johnston, 425 U.S. 219 (1976); Parker v. Flook, 437 U.S. 584 (1978); Diamond v. Diehr, 450 U.S. 175 (1981).

104. *Manifesto, supra* note 6, at 2345.

105. Brian Kahin, *The Software Patent Crisis,* TECH. REV., April 1990, at 53-58 (noting that an attempt to patent the behavior of computer programs may provide too much protection).

106. *See* 17 U.S.C. §§ 901-06 (1984) (describing the Semi-Conductor Chip Protection Act).

107. *See Manifesto, supra* note 6, at 2346. The authors here demonstrate how the patent process, which is not suitable to semiconductors, requires the applicant to make an inventive advance over the prior art. *Id.*

108. *Manifesto, supra* note 6, at 2346-47 (reiterating the resemblance of semiconductor chip and software inventions patterns, and Congress' need to provide the same protection for software and semiconductors to dispel the uncertainty for the scope of protection available).

109. *See* Note, *supra* note 18, at 1049 (blaming the inability of patent law to accommodate abstract invention, through legal protection that treats hardware, software, and algorithms in radically different ways).

110. Burke, *supra* note 12, at 1124. The application process to obtain a patent for computer software is difficult. Burke, *supra* note 12, at 1124. The burden of proof for showing best mode and enablement, as well as the complete lack of prior art in the case of computer programs makes the process unprofitable and too expensive for small firms. Burke, *supra* note 12, at 1124.

den hose to put out a three-alarm fire: the hose may help a bit, but really does nothing to solve the major problem.

C. A PROPOSAL

"The law must be stable, but it must not stand still."[111]

Instead of attempting to pigeonhole computer software within the purview of either patent or copyright protection, why not simply cut another hole? "Hole cutters," proposing a *sui generis* approach to software protection, are met with well wishers[112]and nay-sayers alike.[113] Conversely, the courts' attempt to pigeonhole software into patent or copyright law creates confusion.[114] The courts simply cannot apply century-old copyright[115] or patent[116] standards to a twentieth and twenty-first century technology, while also leaving important software elements unprotected.[117] The pioneers of intellectual property protection never anticipated encountering an animal with the voracity of software.

---

111. JOHN BARTLETT, BARTLETT'S FAMILIAR QUOTATIONS 731 (15th ed. 1980) (quoting Roscoe Pound, *Introduction to the Philosophy of Law*).

112. *See, e.g.*, Robert A. Arean, *A Proposal for the International Intellectual Property Protection of Computer Software*, 14 U. PA. J. INT'L BUS. L. 213, 232-42 (1993) (arguing for *sui generis* protection due to the shortcomings of copyright law); S. Carran Daughtrey, *Reverse Engineering of Software for Interoperability and Analysis*, 47 VAND. L. REV. 145, 183-87 (1994) (discussing the shortcomings of copyright and the need for *sui generis* protection); John C. Phillips, *Sui Generis Intellectual Property Protection for Computer Software*, 60 GEO. WASH. L. REV. 997, 1032-41 (1992) (arguing against the current copyright protection in favor of a specific *sui generis* legislative scheme); and *Manifesto*, *supra* note 6 (advocating a market-reflective *sui generis* software protection scheme).

113. *See, e.g.*, 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 (1994) (arguing that software protection is possible through the exclusive use of copyright protection); David Abraham, *Suggestions for Improved Intellectual Property Protection of Software, Or Where is Alexander When You Really Need Him?*, 23 S.U. L. REV. 293, 305 (1995) (arguing that if the Patent Act, 35 U.S.C. § 101, was literally applied to software and the algorithm restriction was lifted, there would be no need for increased or *sui generis* protection).

114. *See Altai*, 982 F.2d 693, at 712. The *Altai* court explicitly states, "to be frank, the exact contours of copyright protection for non-literal program structure are not completely clear." *Id.*

115. The courts still cite *Baker v. Selden* in an attempt to consider whether a particular element of software should have patent or copyright protection. 101 U.S. 99 (1879).

116. Nichols v. Universal Pictures Corp., 45 F.2d 118 (2d Cir. 1930) (presenting the a-f-c test for the first time).

117. The courts deciding software copyright cases utilize Judge Learned Hand's abstraction-filtration-comparison test, or a derivation thereof, in a vein attempt to determine how to copyright and protect computer software. The courts still appear to struggle with the application of the a-f-c test to computer software. *See supra* notes 42-43 (defining the a-f-c test).

1. *The Current and Potential Configuration of the Software Industry*

The computer industry understands the figurative and monetary value of software. In doing so, the computer industry has shifted most of its resources to the research, the production, and the marketing of software.[118] The relatively new software patent increases the overall value of computer software to the industry, encouraging innovation by the "big three"[119] and small businesses alike.[120] This increased value of software comes with a price in the form of decreased competition and reduced entry in the software market.[121] Thus, in order to protect the software industry from becoming an air-tight oligopoly like the auto industry in the 1950's,[122] the law must provide greater protection for the ideas and innovations of the smaller companies.[123]

2. *Time is of the Essence*

The fast pace of incremental innovations of the computer industry, where any computer or piece of software purchased today is obsolete six months later, dictates the need of protection that is flexible, short-lived, high-powered, easily obtained, and quickly-implemented as soon as pos-

---

118. *See* Burke, *supra* note 12, at 1120. This great shift in the industry is mostly due to: (1) the shift from mainframes and mid-range computers to micro-computers and workstations (due to the increased speed and technology of semiconductors); (2) the shift to open systems from more profitable proprietary systems; (3) the increased commodization of many products; and (4) the greater competition from foreign producers, especially those in the Far East and Asia. Burke, *supra* note 12, at 1120. The software industry now has more than twice the number of employees as the hardware industry. Burke, *supra* note 12, at 1120.

119. *See* Lawrence M. Fisher, *Novell to Acquire WordPerfect*, N.Y. TIMES, Mar. 22, 1994 , at D1. Similar to the auto industry, the three software manufacturers of Microsoft, Lotus and Novell, through recent takeovers, control 89% of the word processing market and 97% of the spreadsheet market for personal computers. *See also* G. Pascal Zachary, *Consolidation Sweeps the Software Industry; Small Firms Imperiled*, WALL ST. J., Mar. 23, 1994, at A1.

120. *See* Burke, *supra* note 12, at 1124.

121. Fisher, *supra* note 118. Fisher warns that the large computer firms, the "big three" are slowly devouring the small software firms. Fisher, *supra* note 118. With the increased protection afforded by the software patent, coupled with the ease of copying ideas in the software industry and difficulty of small companies to obtain software patents, the new patent is decreasing the competition of the market by closing it to small firms. Fisher, *supra* note 118.

122. *See* Robert X. Cringely, *If Novell is Ford and Lotus is Chrysler, Does That Make Borland Hudson?*, INFOWORLD, Mar 28, 1994 at 98 (comparing the market structure of software industry and the auto industry). *See also* Apple Computer v. Microsoft, 799 F. Supp 1006 (N.D. Cal. 1992), *aff'd*, 35 F.3d 1435 (9th Cir. 1994).

123. *See* Burke, *supra* note 12, at 1130-31 (noting that antitrust laws are in existence to regulate any anti-competitive behavior, but patent law is needed to see that new entrants in the market have protection of their innovations).

sible.[124] One wonders why copyrights and patents provide protection to a piece of software for seventy-five years[125] and twenty years,[126] respectively, when the software in question is most likely out of date within three to five years.[127] Also, the amazing exponential growth of the computer industry exposes a problem in our legal system, namely that the effort needed to start the proverbial wheels of justice to turn may pale in comparison to the rewards.[128] Hence, any changes made to the current system of patents must start now and the changes must incorporate a vision toward the "future."

3. *The Smorgasbord Approach*

Lawmakers and lawyers alike understandably despise any change in the legal system because of the fear of the unknown, and the effort required to familiarize themselves with new laws and the ramifications thereof. To ease the lawyer's pain and cost of learning a completely new system, new software protection must take as much of the material and policy from the positive aspects of the current law in a smorgasbord approach: *take a little from everything, but only take what you like.* Using an economics rationale to decrease the "transaction costs"[129] of imple-

---

124. *See* Cringley, *supra* note 121, at 98. In further comparing the software industry to the auto industry, Cringley notes that the what took the auto industry 50 years to evolve, the computer software industry did in ten. Cringley, *supra* note 121, at 98. Thus, if new intellectual protection is not provided to spur and protect the innovation of the small firms, the country must deal with a tight oligopoly, instead of an efficiently competitive market. Cringley, *supra* note 121, at 99.

125. 17 U.S.C. § 102(a) (1988 & Supp. 1990).

126. 35 U.S.C. § 154(a)(2) (1994). Congress and the Patent and Trademark Office just increased the time period for patent protection from seventeen to twenty years. *Compare* 17 U.S.C. § 102(a) (1988 & Supp. 1990) *with* 35 U.S.C. § 154(a)(2) (1994).

127. ROBERT L. MILGRIM, MILGRIM OF LICENSING § 5.00 at 5-7 (3rd ed. 1992) (stating that the practitioner can safely assume that the subject matter protected will rapidly expand in any industry where electronics or computers play a role).

128. Congress took over twenty years to realize that the microchip deserves special protection because it is different from any other subject matter encountered in intellectual property. *See* 17 U.S.C. §§ 901-06 (1984) (citing the Semi-Conductor Chip Protection Act). Software and semiconductors demonstrated similar leaps in technology. *Id.*

129. ROBERT COOTER & Thomas Ulen, Law and Economics 84 (2d ed., Addison-Wesley 1992). In economic terms, transaction costs are costs of exchange. *Id.* The effort of finding a partner with whom to exchange, negotiating the exchange, enforcing the exchange, and gaining the information to do each costs time, effort and money. *Id.* These costs of effort, time and money in the exchange process are considered transaction costs. *Id.*

The landmark Coase theorem states that without these transaction costs impeding efficient bargaining, two people will always bargain to an efficient end. *Id.* at 82. In the system that the *Manifesto* or any *sui generis* proponent proposes, the cost of monitoring the market and obtaining the information needed to implement their system is very great. *Manifesto*, *supra* note 6, at 2401-15. Also, the costs of strategy with the onslaught of new lawsuits under a new system, requiring more time for the attorney to learn and borne by

menting a new and unknown system into law, the new system should utilize as many aspects relating to the current system as possible.[130] Therefore, a new system must combine the slight scope[131] and ease of obtainability[132] of a copyright with the increased protection and force of efficiency under the "best mode"[133] of a patent. The system should be motivated toward protecting the valuable element of software, the behavior, and not the text or nonliteral elements[134] of the software.[135]

4. *The West Germans Do It Better*

The Federal Republic of Germany created a two-tiered system of patents where major inventions and innovations receive full-term patents, while minor inventions receive petty patents, for a shorter duration.[136]

---

the client, are detrimental for efficient bargaining under the Coase theorem. COOTER & ULEN at 83.

The Coase theorem corollary underlines the need for efficient property rights. *Id.* at 82. The Coase Corollary states: "When transaction costs are high enough to prevent bargaining, the efficient use of resources will depend upon how property rights are assigned." *Id.* In the reality of our litigious society, transaction costs are seldomly considered "low," even if utilizing a smorgasbord rationale to decrease the transaction costs of the legal system. Ergo, according to the Coase corollary, implementing a system where property rights provide an efficient outcome is essential to overall economic efficiency. *Id.*

130. *See* Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUD. 321, 372 (1995) (noting that the major economic problem with a *sui generis* approach to property rights has to do with the uncertainty, requiring inefficient negotiation and litigation to completely define the new property rights).

131. *See* Stern, *supra* note 80, at 1246-47 (noting the efficiently small scope of protection that a copyright provides, in comparison to other forms of protection such as a patent or trademark).

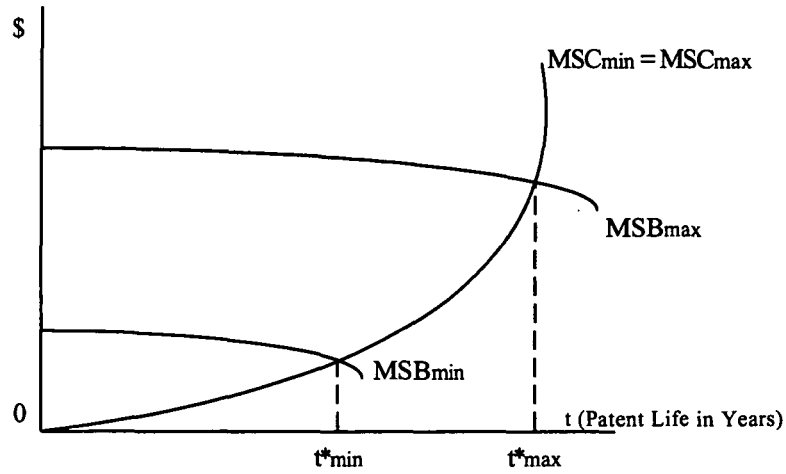
132. *See* Amin, *supra* note 82, at 22 (highlighting the cost of obtaining a software patent, in comparison to the dramatically lower cost of obtaining a copyright for software).

133. *See* Graham & Zerbe, *supra* note 100, at 96 (noting that the best mode disclosure, as well as the enablement disclosure, provides the market place with an efficient free-flow of information in order to build on previous innovations as quickly as possible).

134. *See supra* notes 42-43 (outlining the struggle to find an adequate system to separate and protect the nonliteral elements of computer software).

135. The best and most far-reaching aspect of the *Manifesto* is the pinpointing of the value-giving element of software, the behavior. *See Manifesto*, *supra* note 6, at 2379. The *Manifesto* correctly realizes that lawmakers must stray from the various versions of the a-f-c test and divide software into elements easily defined, elements which are based upon the valuable behavior of the software. *Manifesto*, *supra* note 6, at 2420.

136. HERMANN KINKELDEY & WILFRIED STOCKMAIR, THE NEW GERMAN PATENT LAW 33 (Hermann Kinkeldey & Wilfried Stockmair, trans., Verlag Chemie 1981) (translating § 17 of the West German version of the Patent Act as it pertains to major and minor patents).



A multi-tiered system is more economically efficient than the current United States patent system, especially for industries that traditionally innovate incrementally.[137] By applying the German multi-

137. COOTER & ULEN, LAW AND ECONOMICS 136 (1st ed., Harper Collins 1988). From an economic standpoint, the social benefit conferred to the consumer by inventing the new invention must take into account the value to society of the invention. *Id.* Likewise, this invention must also adequately compensate through a social cost burdened to the consumers who purchase this invention. *Id.* Without adequate competition, there is no incentive for innovators to invent, decreasing the overall social value, technological level, and market efficiency of the market. *Id.*

Under the current system, the social benefit of a twenty year software patent or a 75 year computer copyright is not high enough to justify the high social cost of withholding or licensing this information or method to the rest of the computer industry. By decreasing the time of protection provided to software, the social benefit will equal the social cost of the invention. ROBERT D. COOTER & THOMAS S. ULEN, LAW AND ECONOMICS ch. 5, at 20 (1994) (unpublished manuscript, on file with author).

Assume, arguendo that in figure 1 above, the vertical x-axis is social cost in dollars and the horizontal y-axis is the patent life in years, where $t^*_{max} > 20 > t^*_{min}$. It follows that if two inventions are given a 20-year patent, society will have given too much protection to the minor improvement in technology and too little protection to the major invention. COOTER & ULEN, *supra* note 128, at 123. This results in adverse social consequences. COOTER & ULEN, *supra* note 128, at 123. For example, if the 17-year patent gives a greater return to the minor improvement than the major invention, inventive sources may flow toward minor inventions and away from major inventions. COOTER & ULEN, *supra* note 128, at 124.

A single 20-year patent cannot balance the social benefits of inventive activity against the social costs of limited access. COOTER & ULEN, *supra* note 128, at 124. In the case of a multi-tiered patent for software, there are six different levels of marginal social benefit based upon six elements of software value. If graphically represented, the patent proposal has six marginal social benefit ("MSB") curves and accordingly six possible patent lives in an attempt to best disperse the marginal social cost ("MSC") of a patent monopoly to society. Likewise, a multi-tiered system reduces the value of patent to the owner at a greater rate than a single term patent system, thereby reducing the amount of resources devoted to obtaining patents. RICHARD A. POSNER, ECONOMIC ANALYSIS OF LAW 39 (1992).

tiered system, graphically depicted above, to software, where the more valuable elements of the software receive major patents and less valuable elements of the software receive minor patents, the incremental innovations in the software industry are efficiently protected.[138]

Instead of separating software into major and minor patents as in Germany, the length and quality of the patent is dictated by the *Manifesto's* separation of the various elements of software according their utilitarian value.[139] For example, the industrial design of the program behavior receives the patent for the longest duration (approximately ten years)[140] because, according to the *Manifesto*, this aspect of the program is the most valuable.[141] Likewise, the algorithm of the computer program receives the shortest duration (approximately one year)[142] of patent protection because, according to the *Manifesto*, the algorithm is the least valuable element of the computer program.[143]

---

138. POSNER, *supra* note 136, at 39-40.

139. *Manifesto*, *supra* note 6, at 2379. The *Manifesto* divides software into five categories based upon functionality, marketability and valuable behavior: (1) program compilations as a whole or industrial design of the program behavior; (2) subcompilations/subsets of program behavior; (3) features as coherent units of program behavior; (4) program/object code; and (5) algorithms. *Manifesto*, *supra* note 6, at 2379. These categories are in order from most functional, useful, and valuable to least functional, useful, and valuable. *Manifesto*, *supra* note 6, at 2379

The *Manifesto* correctly observes that software manufacturers should have the means to protect the behavior of software; and in doing so, divide software into various elements, according to their respective behavior value and usefulness to the consumer. *Manifesto*, *supra* note 6, at 2379. However, their market approach to protection appears untenable because: (1) lawyers and legal scholars will meet any completely new *sui generis* means of protection with the most resistance; (2) the market research and *a priori* approach that they propose relies on economic and market structure information that is very costly or impossible to obtain; and (3) even if the market information they require was available or cost-efficient, the market for computer software and thus the law under their system changes too quickly. *See* COOTER & ULEN, *supra* note 128, at 136-37.

140. The length of the various levels of patents requires some research into the social cost and benefit of the computer software, similar to the economic analysis above. *See supra* note 128. Regardless, the highest levels of protection cannot remain at the current twenty year level due to the pace of overall technological improvement in the computer industry.

141. The categories mentioned are in order of most to least valuable, and therefore in order of patent strata level of longest duration to shortest duration. *See supra* note 138 (regarding the categories mandated by the *Manifesto*).

142. *Manifesto*, *supra* note 6, at 2379 (regarding the categories mandated by the *Manifesto*). The correct economic analysis must be used to obtain the optimum tenure for protection of all the strata of protection. *See* COOTER & ULEN, *supra* note 128, at 82.

143. Abraham, *supra* note 112, at 303. Some have concerns about the protecting of algorithms by patent law. Abraham, *supra* note 112, at 303. Some argue that algorithms should not even be protected at all, citing the traditional "law of nature" or mathematical exception of patent law. Abraham, *supra* note 112, at 303. Abraham solves the problem of indexing the prior art of algorithms by proposing a new system that indexes the algorithm prior art according to general categories of operation. Abraham, *supra* note 112, at 304.

Furthermore, the scope[144] and ease of obtainability of the patent varies with the respective length of the patent. [145] For example, the standard of proof of novelty, usefulness, and non-obviousness for the one year patent of an algorithm is minute in comparison to the standard for a ten year patent on the industrial design of software. In addition, since the proposed levels of patents are much shorter in duration, the patent application process decreases with respect to time and expense.

5. *The Advantages and Disadvantages of Other Sui Generis Theories*

Other proponents of a *sui generis* revamping of the current protection system for software rely too heavily upon obtaining *a priori* market and industry information, requiring a Ph.D. in economics to successfully apply their respective approaches.[146] Furthermore, most of the recent proponents of *sui generis* protection for computer software agree that if the new protection is to take place, the protection should be in the form of a patent.[147] Other proponents of new software protection believe that the time allotted for intellectual property protection should be re-

---

Abraham also solves the problem of "holding up" the industry when one protects valuable new algorithms when attempting to licensing it by arguing that the rest of market will attempt to invent around the patented algorithm. Abraham, *supra* note 112, at 304. Another solution he proposes contends that cross-licensing of new technology would take place during the wait between disclosure of the algorithm and the actual licensing of the algorithm. Abraham, *supra* note 112, at 305.

144. Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope,* 90 COLUM. L. REV. 839, 890-95 (1990). The article notes that an inventor experiences difficulty in obtaining or granting a patent because of the vast and rigid scope of the current patent law. *Id.* The authors argue that by decreasing the scope of the patent in proportion to the importance of the invention, difficulty in obtaining a patent decreases and economic efficiency increases. *Id.*

145. *Id.*

146. *See, e.g., Manifesto, supra* note 6. The *Manifesto* makes a valiant attempt at pure market-oriented approach to software protection. However, the theory relies upon too much information that is either impossible to obtain until all of the software takeovers occurring in 1994 leave their effects; or too expensive and impractical to obtain. COOTER & ULEN, *supra* note 136. Moreover, the intense economic policies within and surrounding the *Manifesto's* proposal would send all intellectual property attorneys back to school for additional economics classes. Dam, *supra* note 129, at 373.

147. *See generally* Graham & Zerbe, *supra* note 100 (arguing that increased protection in patents and reverse engineering are economically efficient and that reverse engineering can be accomplished within the structure of the Patent Act); Luettgen, *supra* note 7 (arguing that the usefulness of the nonliteral aspects of computer software is not adequately protected by copyright); Abraham, *supra* note 112 (using a literal interpretation of the patent act (35 U.S.C. § 101), he argues that patents can and should protect computer software); Paley, *supra* note 30 (proposing to amend the current patent act to include algorithms, thereby appending the Software Act to the Patent Act rather than the Copyright Act).

duced.[148] However, none combine the two notions, or propose protection based upon a West German smorgasbord rationale.

Within the current system, utilizing the smorgasbord approach, the high protection afforded by a patent appears to be the only means software's utilitarian value through which computer companies can efficiently protect the valuable behavior and functionality of the software.[149] The proposed multi-tiered system provides a copyright-like quality by limiting the scope of certain patents, thus easing the requirements to obtain a patent.[150] The new system solves the paradox of the Patent and Trademark Office related to prior art by allowing the office to issue more patents. Also, the proposed system facilitates the efficient free exchange of information by keeping the Patent Act's section 112 disclosure[151] already embedded in the system.

IV. CONCLUSION

A multi-tiered system of patents is more efficient for both the economy and society.[152] The proposed multi-tiered patent system is not necessarily *sui generis* because the efficiency gained through a multi-tiered patent system is not limited to the computer software industry.[153] An excellent example of an industry that would greatly benefit from a multi-

148. Amin, *supra* note 82, at 24 (arguing that the scope of protection should be reduced to one or two years); Douglas J. Masson, Note, *Fixation on Fixation: Why Imposing Old Copyright Law on New Technology Will Not Work*, 71 IND. L.J. 1049 (1996) (arguing that an author of a copyright protected work should have protection for an arbitrary amount of five years).

149. Abraham, *supra* note 112, at 304. Abraham gives great deference to the *Manifesto* because the *Manifesto* acknowledges the diversity of program and protects software by dividing software into five categories. Abraham, *supra* note 112, at 304. Abraham then argues that by literally interpreting the Patent Act of 35 U.S.C. § 101, the algorithm exclusion of the Patent Act will be abolished. Abraham, *supra* note 112, at 305. This abolishment, argues Abraham, can adequately protect computer software. Abraham, *supra* note 112, at 305. However, Abraham's plan does not solve the shortcomings of patents, namely in the fast-moving field of computers, the difficulty and the high cost of obtaining a patent, which prices the small firms out of the market. Abraham, *supra* note 112, at 305.

150. If a patent is for a shorter duration, the PTO is less concerned and therefore less restrictive with the novel, non-obvious, useful and best mode requirements. *See* Stern, *supra* notes 80 and 130 (noting that the wide scope of patents causes conservative issuing of patents by the PTO). Under the current system, these requirements are strictly adhered to, which denies protection to small firms. *See supra* note 93 and accompanying text (reiterating the strict standards for obtaining patent protection in regards to software).

151. *See supra* note 56 (describing disclosure requirements and enablement guidelines for software patents).

152. *See supra* note 127 (explaining the greater efficiency of a multi-tiered patent system over the current single-tiered patent system).

153. The two-tiered system in West Germany is applied to any invention that qualifies for a patent. COOTER & ULEN, *supra* note 136, at 136.

tiered patent system is the drug manufacturing industry.[154] Any industry that produces patentable products where the innovation is research-intensive and incremental in nature, like the software industry, benefits economically from a multi-tiered system.[155]

Obviously, the current patent system cannot adequately protect the functionality and behavior of software.[156] In attempting to perform the much-needed revamping of the current intellectual property system, any new or *sui generis* proposal must utilize as much legal theory and policy from the current system as possible. This "smorgasbord" approach to intellectual property reform eases the difficulty of learning and implementing a completely new system, thereby decreasing transaction and retraining costs.[157] Because the minor patents in the multi-tiered system are easier to receive and cost less to obtain, small firms, previously unable to compete in the software market, can now compete. The implementation of a multi-tiered patent system increases the overall efficiency of virtually any market that the lawmakers deem applicable. In the face of competitive market destruction, the lawmakers must force the law to keep pace with technology.

Larry N. Woodard

154. Burke, *supra* note 12, at 1129. The current patent system innately spurs innovation and dissemination of new technologies by granting property rights to original inventors. Burke, *supra* note 12, at 1129. U.S. drug companies spend billions on research and development, and once the drug is discovered and marketed, the drug can be copied at little cost. Burke, *supra* note 12, at 1129. Moreover, the drug industry rarely makes quantum leaps in innovation, but rather small steps. Burke, *supra* note 12, at 1129. A multi-tiered system can spur innovation by increasing protection for those easily copied elements of a small step in innovation that would otherwise not hold a patent. Burke, *supra* note 12, at 1129. *See also* Merges & Nelson, *supra* note 143, at 847 (showing the difficulty that the drug industry undergoes in obtaining patents for incremental innovations).

155. Any industry that invests heavily in research and development and innovates with small steps instead of large leaps has the same economic analysis as the software industry. *See supra* notes 127-28 (describing software industry multi-tiered economic analysis).

156. *See supra* notes 125-27 and accompanying text (proving that the current patent system cannot protect software behavior).

157. Recall, that economic efficiency increases and rational people tend to negotiate to efficient outcomes as the transaction costs decrease. *See supra* note 128 (defining the Coase theorem and transaction costs).