Vanderbilt Law Review

Volume 37
Issue 1 *Issue 1 - January 1984*

Article 3

1-1984

The Policy Implications of Granting Patent Protection to Computer Software: An Economic Analysis

Jeffrey S. Goodman

NOTE

The Policy Implications of Granting Patent Protection to Computer Software: An Economic Analysis*

* Cited in Paine, Webber, Jackson & Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358 (D. Del. 1983) (Latchum, C.J.).

I. INTRODUCTION

Computer software technology commands widespread public attention in contemporary society because the application and use of computers has branched into most areas of daily life.[1] The demand for software programs has burgeoned in part because computer hardware such as complicated computer circuitry has become very affordable to the general public.[2]

Thus, computer software programmers now enjoy a broadly based commercial market for their software inventions. Unfortunately, software competitors can appropriate software innovations very easily. Software programmers, therefore, looking for some way

1. *See, e.g.*, SCI. AM., Sept. 1982 (Scientific American magazine devoted its entire September issue to mechanization of work through the use of computers. It documented the role of computers in such areas as design, manufacturing, commerce, and office work.). *See* Branscomb, *Bringing Computing to People*, COMPUTER July 1982, at 68. Large grocery stores, for example, use computer assisted electronic cash registers coupled to universal product code scanners and weighing scales to assist cashiers in grocery store check-out lines. These computer assisted devices help grocers expedite sales of goods, preserve complete records of all transactions, and coordinate the aggregation of all sales records to produce inventory and other business documents at the end of business periods. Ernst, *The Mechanization of Commerce*, 247 SCI. AM., Sept. 1982 at 132, 140.

2. "The most startling change in computer system technology has been the dramatic decrease of hardware cost by a factor of 2 every two to three years since 1945." P. WEGNER, INTRODUCTION AND OVERVIEW, RESEARCH DIRECTIONS IN SOFTWARE TECHNOLOGY 2 (P. Wegner, ed. 1980). *See also* C. SIPPL & R. SIPPL, COMPUTER DICTIONARY AND HANDBOOK 919, 920 (1980) (Appendix N: The Progress, Impact, and Future of Computers). Several different theories have arisen concerning the relationship of computer software and hardware. One theory contends that the "program is basically a set of instructions that sets a computer's switches so that it can perform a particular function." Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies and Computer Programs*, 84 HARV. L. REV. 281, 340-41 (1970); *see also* Davis, *Computer Programs and Subject Matter Patentability*, 6 RUT. J. COMPUTERS, TECH. AND THE LAW 1 n.3 (1977); REPORT OF THE PRESIDENT'S COMMISSION ON THE PATENT SYSTEM, TO PROMOTE THE PROGRESS OF . . . USEFUL ARTS IN AN AGE OF EXPLODING TECHNOLOGY 12 (1966). A second theory views software programs as adding the control function to an otherwise incomplete piece of computer bardware. Gemignani, *Legal Protection for Computer Software: The View from 1979*, 7 RUT. J. COMPUTERS, TECH. AND THE LAW 269, 279 & nn.46-48 (1980). Another theory argues tbat computer software resembles a coded message "[b]ecause the program is stored in a computer as data . . .[which] when written out, appears as just a string of 0's and 1's." *Id.* at 280.

to protect their original programs, have sought patent protection to establish legally recognizable and protectable rights in their innovations. The Patent Office and courts, however, consistently have denied applications for patent protection of innovative software programs, reasoning that the computer algorithm, the foundational element of computer software, is an unpatentable scientific principle.

The primary objective of patent laws is to advance the sciences.[3] The patenting of an invention essentially is an exchange between an inventor and society[4]—the inventor receives the exclusive right to make, use, or sell[5] an invention for a certain period of time, and society receives the benefit of using the invention[6] and of receiving full disclosure of the invention's underlying idea.[7] Thus, deciding whether an invention deserves patent protection requires an inquiry into whether the work will deliver benefits to society in excess of the costs that society must bear by granting the inventor exclusive property rights in the invention.[8]

Because a patent is a legal monopoly,[9] the requirements for

---

3. Sinclair & Carroll Co. v. Interchemical Co., 325 U.S. 327, 330-31 (1945); P. JOHNSON, THE ECONOMICS OF INVENTION AND INNOVATION 42 (1975) ("The encouragement of inventive activity . . . is . . . a major argument for the patent system."). *See* F. SCHERER, INDUSTRIAL MARKET STRUCTURE AND ECONOMIC PERFORMANCE 440 (1980).

4. *See* J. PARKER, THE ECONOMICS OF INNOVATION 303 (1978).

5. 35 U.S.C. § 271(a) (1976) provides in pertinent part that: "whoever without authority makes, uses or sells any patented invention . . . infringes the patent."

6. A lively debate persists about a patentee's right to suppress a patent. Courts currently hold that a patentee's "title is exclusive . . . and that he is neither bound to use his discovery himself, nor to permit others to use it." Heaton-Peninsular Button-Fastener Co. v. Eureka Specialty Co., 77 F. 288, 295 (1896), *adopted in* Continental Paper Bag Co. v. Eastern Paper Bag Co., 210 U.S. 405, 425 (1908). Justice Douglas, dissenting in Special Equipment Co. v. Coe, 342 U.S. 370, 380 (1945), recommended abandoning the *Continental Paper Bag* rule and returning to the rule that a patentee "is bound to use the patent himself or allow others to use it on reasonable . . . terms."

7. The specification [of the invention claimed in the patent application] shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise and exact terms as to enable any person skilled in the art to which it pertains . . . to make and use the same.

35 U.S.C. § 112 (1976). *See* 35 U.S.C. §§ 111-15 (1976).

"The function of a patent is to add to the sum of useful knowledge." A & P Tea Co. v. Supermarket Equipment Corp., 340 U.S. 147, 152 (1950). In economic terms, "[p]atents are a piece of social engineering which deliberately support monopoly at the expense of competition on the grounds that the benefits to the community of improving the potential flow of new knowledge, outweigh the misallocation effects associated with deliberately creating market imperfections." J. PARKER, *supra* note 4, at 302.

8. An inventor's exclusive rights under a patent last for seventeen years and give the patentee "the right to exclude others from making, using, or selling the invention . . . ." 35 U.S.C. § 154 (1976).

9. R. POSNER, ECONOMIC ANALYSIS OF LAW 199, 208 (2d ed. 1972).

receiving a patent grant are very strict.[10] The work an inventor seeks to patent must fall within the scope of subject matter that the patent laws protect.[11] This subject matter requirement persists as the major bar to patent law protection of computer software.[12] The Patent Act does not protect natural phenomena, also known as the "laws of nature," and descriptions of these phenomena.[13] The descriptions of algorithms that computer scientists have accepted[14] and the manner in which these scientists have incorporated algorithms into the production of computer programs reveal that algorithms conceptually are not equivalent to these laws of nature.[15] Nevertheless, the Supreme Court consistently has erred in its interpretation of computer software algorithms by equating them with mathematical laws.[16] Consequently, the Court has held software algorithms to be impermissible subject matter for patent protection.[17] Software programs will not qualify as appropriate subject matter for patent protection until the Court changes its characterization of algorithms.

A change in the Court's characterization of algorithms, however, is merely the starting place for an evaluation of the propriety of granting patent protection to computer software programs. Courts also must consider the difficult policy questions concerning whether the patent system should protect innovations in software technology. The patent system is a mechanism that the federal government has sponsored to achieve social welfare goals.[18] Thus,

---

10. To receive patent protection an invention generally must be new, useful, and nonobvious to a person ordinarily skilled in the pertinent art. 35 U.S.C. §§ 102-03 (1976). Courts have construed these requirements strictly. *See* Sakraida v. Ag Pro, Inc., 425 U.S. 273 (1976); Graham v. John Deere Co., 383 U.S. 1 (1966).

11. 35 U.S.C. § 101 (1976) states in pertinent part: "Whoever invents or discovers any new and useful process, machine, manufacture or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor . . . ."

12. *See, e.g.*, Novick & Wallenstein, *The Algorithm and Computer Software Patentability: A Scientific View of a Legal Problem*, 7 RUT. J. COMPUTERS, TECH. AND THE LAW 313, 333 (1980).

13. Gemignani, *supra* note 2, at 294.

14. An algorithm is simply a detailed process for solving a specific problem. *See* T. WALKER, FUNDAMENTALS OF COMPUTER SCIENCE 8 (1975). *See infra* notes 31-37 for a discussion of software algorithms.

15. Novick & Wallenstein, *supra* note 12, at 333-35; Note, *Algorithm Patentability After* Diamond v. Diehr, 15 IND. L. REV. 713, 730 (1982).

16. *See infra* notes 98-174 and accompanying text.

17. *Id.*

18. *Cf.* SUBCOMM. ON PATENTS, TRADEMARKS AND COPYRIGHTS OF THE SENATE COMM. ON THE JUDICIARY, 85th Cong., 2d Sess. 1, STUDY 15; F. MACHLUP, AN ECONOMIC REVIEW OF THE PATENT SYSTEM (1958) ("a patent confers the right to secure the enforcement power of the state in excluding unauthorized persons . . . from making commercial use of a clearly de-

the government must identify and assess the relationship between a new technology and the purposes and policies of the patent system before allowing patents to protect works in this area. Identification and assessment of this relationship requires an inquiry into the relationship between computer technology, public policy, economic theory, and legal doctrine.[19]

This Note analyzes the propriety of granting patent law protection to computer software by viewing this problem from economic, legal, public and technological policy perspectives. Part II explains the relationship between computer hardware and software, discusses the role of algorithms in software development, and traces the development of the computer software industry. Part III analyzes the economic policies underlying the patent system. Part IV identifies the patent law principles that are relevant to the software patentability issue and discusses their underlying policy foundations. Part V examines the Supreme Court's application of these principles in the leading software patent cases and concludes that the Court's failure to understand computer technology has caused it to withhold patent protection from computer software. Part V reveals that the Supreme Court has mischaracterized software algorithms by treating them as unpatentable mathematical laws. Part VI of this Note analyzes the benefits and costs of granting patent protection to computer software and demonstrates the compelling societal need for this protection. Part VI also proposes an addition to the Patent Office of a small staff of computer science experts to remedy the administrative problem of processing software patent applications.

II. COMPUTER SOFTWARE: ALGORITHMS AND THE MARKETPLACE

The outcome of patent litigation, including requests for patent protection of computer software, depends largely upon a judge's understanding of what an inventor claims to have developed.[20] The

---

fined invention."); R. POSNER, *supra* note 9, at 199, 208 (stating that a patent is a legal monopoly); R. MILLER, INTERMEDIATE MICROECONOMICS: THEORY, ISSUES AND APPLICATIONS 304 (1978) (discussing the effect of patents on barriers to entry).

19. These areas of study often are incompatible and difficult to examine. Computer technology is esoteric and not widely understood by people who are not experts in computer science. The public policy considerations underlying the patent laws vacillato between the need to encourage individuals to make discoveries and innovations for public benefit and to reward inventors by providing legal protection for their innovations, and the desire to preserve America's competitive economic system that abhors any form of exclusive market power.

20. Diamond v. Diehr, 450 U.S. 175, 193-94 (1981) (Stevens, J., dissenting). For exam-

Supreme Court's decisions that refuse to grant patent protection to computer software contain three basic misunderstandings. First, the Court improperly limited its definition of algorithms to procedures that solve mathematical problems. Second, not all computer software algorithms are equivalent to unpatentable natural laws. Last, the Court failed to realize that software algorithms satisfy its interpretation of the Patent Act's definition of process.[21] Because of the Supreme Court's confusion, lower tribunals such as the Patent and Trademark Office, and Board of Appeals of the Patent and Trademark Office, and the Court of Customs and Patent Appeals are perpetuating an erroneous view of computer software technology and creating a body of case law that rests upon an improper application of patent principles to computer software programs.

A. Computer Software Technology

Software programs are the heart of computers. They contain logical sequences of instructions[22] that direct computers to perform various tasks.[23] Computer programmers begin designing software by creating flow charts that express the precise method of solving a problem or performing a task. They then translate this flowchart into a language compatible with computers.[24] The process that computer programmers use to create software programs is very complex and costly. For example, the SABRE program that American Airlines used to coordinate passenger flight reservations contained more than one million instructions and cost more than $30 million to develop.[25] The typical cost of developing most software programs, however, ranges between $50,000 and $500,000, with an average cost of about $200,000.[26] Without these costly or complex software programs, however, computers merely are inert assem-

---

ple, one commentator points out that " '[w]hat the Supreme Court believes a [computer] program is, or is not,' determines the kind and extent of legal protection that will be accorded it. Unfortunately, however, the Supreme Court, as well as most other courts, is ill-equipped to deal with the technological complexities of computer programs." Gemignani, *supra* note 2, at 276 & n.32.

21. *See infra* notes 98-174 and accompanying text.

22. Gemignani, *supra* note 2, at 271.

23. This Note's discussion of computers concerns digital computers, which comprise more than 90% of all computers that are currently in operation. *See* C. SIPPL & R. SIPPL, *supra* note 2, at 657.

24. Gemignani, *supra* note 2, at 276.

25. *Id.* at 276 n.36 (citing Burck, *"On Line" in "Real Time,"* FORTUNE, April 1964, at 145).

26. Gemignani, *supra* note 2, at 276 n.36 (citing amicus curiae brief of Applied Data Research at 3-4 & n.7, Parker v. Flook, 437 U.S. 584 (1978)).

blages of electromechanical and electronic modules.[27]

Computer software programs fall into two broad categories: operating systems programs and applications programs. Operating systems programs control the general operation of the computer.[28] Applications programs, in comparison, direct computers—computer machines and the operating systems program—to perform particular tasks.[29] The applications program is the most important software program to typical computer users because it provides the instructions that tell the computer how to solve their problems. The expression of the applications program is an algorithm.[30]

B. Software Algorithms

Computer software is an algorithm or series of algorithms, which are processes that solve problems and perform tasks.[31] All people unwittingly use algorithmic processes to solve most problems they encounter in daily life.[32] People who seek solutions

---

27. "Standing alone, however, a computer is like a piano without music. The 'music' to make a computer 'play' is the computer program. Simply put, a computer program is a plan for automatically solving a problem." Root, *Protecting Computer Software in the '80's: Practical Guidelines for Evolving Needs*, 8 RUT. J. COMPUTERS, TECH. AND THE LAW 205, 207 (1981). "A computer without any programming of any kind is simply . . . a lifeless form incapable of any task." Gemignani, *supra* note 2, at 271.

28. Operating systems programs provide the "[p]lans or instructions for controlling input/output operations, remote data transmissions, and multiple users which can be used and reused to control these operations." C. SIPPL & R. SIPPL, *supra* note 2, at 423-24.

29. Gemignani, *supra* note 2, at 271.

30. T. WALKER, *supra* note 14, at 42-45. Computer programmers frequently express algorithms as a series of statements or as a sequence of commands in a flowchart. *Id. See, e.g.*, Gemignani, *supra* note 2, at 272-73, nn.14 & 16. Computer programmers design algorithms to solve a wide variety of problems of differing complexity. Novick & Wallenstein, *supra* note 12, at 334 n.175. For example, programmers have designed algorithms to solve complicated mathematical problems, to bake cakes, and even to operate a pizza parlor. *Cf.* T. WALKER, *supra* note 14, at 8-14.

31. I. HORABIN & B. LEWIS, ALGORITHMS 15 (1978).

32. An algorithm representing the safe procedure for lifting a cup of coffee off a saucer, for example, might read as follows:

(1) Start;
(2) Place hand lightly on cup;
(3) If cup is not too hot to maintain contact, skip to step (7);
(4) Remove hand from cup;
(5) Cover hand with napkin—this unit is now the "hand";
(6) Return to step (2);
(7) Lift cup slightly off saucer;
(8) If cup does not begin to slip, skip to step (14);
(9) Replace cup on saucer;
(10) Remove hand from cup;
(11) Wipe clean outsides of both hand and cup;

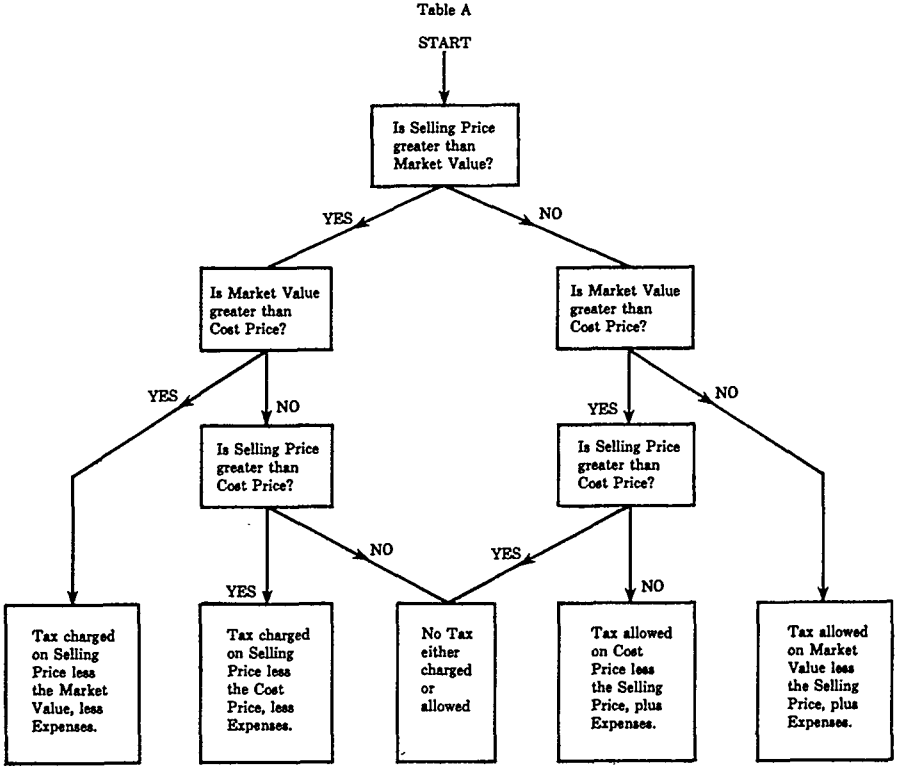
to complex problems, however, frequently choose to employ algorithmic problem solving methods. Lawyers, for example, utilize algorithmic processes in their highly formalized method of researching and structuring legal problems.[33] The Court's confusion about the use of algorithms in computer science, however, stems

---

(12) Place hand on cup;
(13) Return to step (7);
(14) Lift cup to desired height;
(15) FINISH.

Novick & Wallenstein, *supra* note 12, at 334 n.175. A retail merchant could use an algorithmic process, like the one that follows, to help it calculate income taxes for income that it earned from selling goods at certain prices:

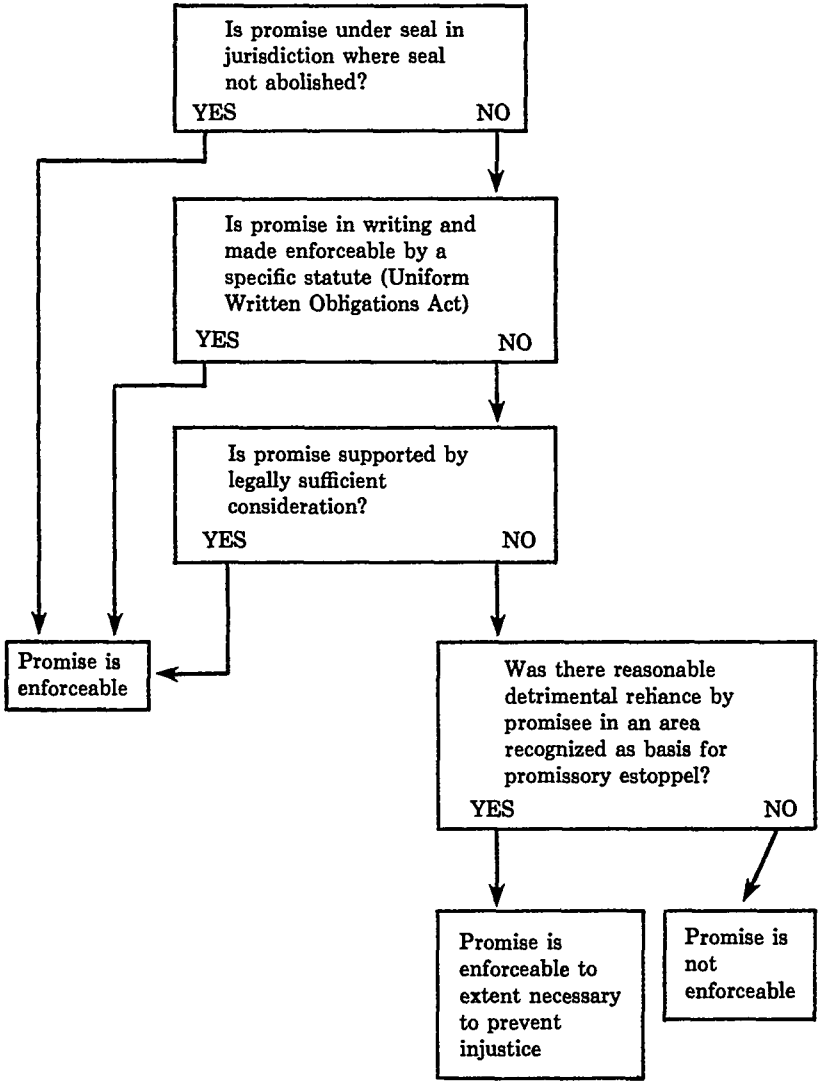
Table A



I. HORABIN & B. LEWIS, *supra* note 31, at 6. See *id.* at 8-13 for examples of uses for algorithms in governmental agencies; pharmaceutical and chemical companies; banks, insurance, and finance companies; manufacturing companies; and universities.

33. *See, e.g.*, Rickert, *Algorithms as an Approach to Learning*, 12 JURIMETRICS J. 171 (1971). For example, the following algorithm illustrates the reasoning process a lawyer might use in determining whether a promise is enforceable:

from the the historical association of the term algorithm with theoretical problem solving in mathematics.[34] This confusion has



Id. at 173.

34. In mathematics the contemporary meaning of the term algorithm originated in 1951 when the Russian mathematician Markov used "the word as a name for his own theory of *effectively computable functions.*" Anderson, *Algorithm* in ENCYCLOPEDIA OF COMPUTER

caused the Court to equate software algorithms with unpatentable mathematical formulas.[35] Confusion about the meaning of the term "algorithm"[36] has been a major reason why the Supreme Court has refused to extend patent protection to software programs.[37]

C. The Computer Software Industry

Until recently only a few companies specialized in producing software products. Before 1970[38] computer hardware manufacturers sold software with hardware as a complete computer package rather than marketing software as a separate product.[39] During this time, typical computer purchasers did not have the requisite knowledge to produce their own software, and unsophisticated purchasers often did not perceive a distinction between software and hardware. Presently, however, most consumers view software and hardware as separate products.

Computer manufacturers today are producing more, better, and less expensive hardware products than ever before. In addition, consumer demand for applications software programs[40] also has grown significantly.[41] Two general types of applications software exist: contract programs, which programmers tailor to the needs of a single user, and general software packages, which manufacturers produce for the public in general. Most software develop-

---

SCIENCE AND TECHNOLOGY 365 (J. Belzer, A. Holzman, & A. Kent eds. 1975) (emphasis in original) (citing Markov, *Teoria Algorifmov* (The Theory of Algorithms), 38 TRUDY MATEMATICESKOGO INSTITUTA IMENI V.A. STEKLOVA 176-89 (1951)). Subsequently, mathematicians have defined algorithm as any recursive computational procedure. *See* Uspensky & Semenov, *What Are the Gains of the Theory of Algorithms: Basic Developments Connected with the Concept of Algorithm and Its Application in Mathematics*, in ALGORITHMS in MODERN MATHEMATICS AND COMPUTER SCIENCE 100-234 (A. Ershov & D. Knuth eds. 1981).

35. *See infra* notes 98-156 and accompanying text.

36. The term "algorithm" derives its origin from the ninth century Arab mathematician Al-Khorezmi whose writings have won him recognition as the father of algebra. *See* Zemanek, *Al-Khorezmi: His Background, His Personality, His Work and His Influence*, in ALGORITHMS IN MODERN MATHEMATICS AND COMPUTER SCIENCE 1-81 (A. Ershov & D. Knuth eds. 1981). In medieval times the word "algorithm" referred to "Arabic numerals, and mathematical computations therewith . . . ." This use of the word gradually vanished. Anderson, *Algorithm*, in ENCYCLOPEDIA OF COMPUTER SCIENCE AND TECHNOLOGY (J. Belzer, A. Holzman & A. Kent eds. 1975).

37. *See* Parker v. Flook, 437 U.S. 584 (1978); Gottschalk v. Benson, 409 U.S. 63 (1972). *See infra* notes 98-133 and accompanying text for a discussion of *Benson* and *Flook*.

38. Gemignani, *supra* note 2, at 274 & n.21.

39. *Id.*

40. *See supra* notes 29-30 and accompanying text.

41. INFOSYSTEMS, July 1978, at 86.

ers[42] annually produce contract and general applications programs in approximately equal numbers.[43] Although individual consumers typically pay for the cost of producing custom-made contract programs, the individual computer companies absorb the costs of producing and marketing general software packages.

The computer software industry has experienced phenomenal growth, but software companies have not developed new software technology quickly enough to keep pace with the burgeoning demand for their products.[44] This anomaly has occurred, at least in part, because new software production requires large investments of time and money[45] to develop programs that competitors unfortunately can appropriate easily.[46] Thus far, software producers primarily have relied upon the trade secrecy laws to combat piracy of their software innovations.[47] This method,[48] however, suffers from two major drawbacks: first, it depends upon the maintenance of secrecy and causes companies to hoard knowledge,[49] and second, its effectiveness is questionable because the requirement of secrecy inhibits companies from aggressively marketing their products to maximize profits from their ingenuity.[50] Hence, software manufacturers continue to seek patent protection for their products.

---

42. The "typical" software company "is independently owned and less than 10 years old. It has fewer than 100 employees, annual sales under $5 million and spends under $100,000 per year on research and development." HARBRIDGE HOUSE, INC., LEGAL PROTECTION OF COMPUTER SOFTWARE: AN INDUSTRIAL SURVEY, *reprinted in CONTU: The Future of Information Technology*, in 4 COPYRIGHT, CONGRESS AND TECHNOLOGY: THE PUBLIC RECORD 355 (N. Henry ed. 1980).

43. *Id.*

44. Note, *Anomaly in the Patent System: The Uncertain Status of Computer Software*, 8 RUT. COMPUTERS, TECH. AND THE LAW 273, 277 (1981) (citing BUS. WK., Sept. 1, 1980, at 46).

45. *See* Root, *supra* note 27, at 209.

46. "The fact that software is a logical . . . product means that costs are concentrated in development . . . [and] there is negligible cost in producing copies of software once it is developed." WEGNER, *supra* note 2, at 4. "It is really amazing how many good, honest citizens who are kind to children and pets, help senior citizens across the street, and wouldn't think of keeping the extra nickel in change that the newsdealer gave them by mistake are perfectly willing to steal software." Rodgers, *The Great Software Ripoff*, POPULAR ELECTRONICS, Nov. 1980, at 4.

47. *See* Gemignani, *supra* note 2, at 304; Keplinger, *Computer Software—Its Nature and Its Protection*, 30 EMORY L.J. 483, 492-93 (1981); Nimtz, *Development of the Law of Computer Software Protection*, 61 J. PAT. OFF. SOC'Y 3, 19 (1979); Root, *supra* note 27, at 225-30; Recent Development, *Protection of Intellectual Property Rights in Computers and Computer Programs*, 9 PEPP. L. REV. 547, 563 (1982); RESTATEMENT OF TORTS § 757 (1939).

48. Some software manufacturers also feel that the protection copyright laws give their innovations is inadequate. Gemignani, *supra* note 2, at 291-92.

49. Note, *supra* note 44, at 280.

50. *See supra* note 46.

III. Patent Economics: Barriers to Entry and the Expectational Effect

The Court, by deciding not to extend patent protection to software programs, has refused to admit that software innovations are identical to scientific processes that historically have received patent protection. As a result, the patent system has not functioned to promote the advancement of software technology. Furthermore, because software creators do not enjoy a legally protected property interest in their software innovations under the patent laws, they lack adequate protection[51] against appropriation of their innovations. The threat of software appropriation discourages software producers from spending time and money to create new software programs because it destroys the producers' ability to recoup costs of production and to profit from their creations.[52] The risk of appropriation is particularly great in the software industry because software innovators basically create knowledge when they develop a new software program, and knowledge is very susceptible to theft. Patent protection of software programs would give innovators a right to prohibit rival software companies from copying and producing their programs. Moreover, the availability of monopoly rights to a software innovation would give software creators the incentive to incur the tremendous cost of creating software programs.

A. Knowledge as a Product

Software programmers essentially create knowledge when they develop a new software program. Programmers' decisions to spend time and money to develop an innovative software program depend largely upon the potential of recovering the costs of their developmental efforts.[53] Thus, programmers price their innovations to reflect these costs. They recognize, however, that knowledge "is expensive to produce, cheap to reproduce, and difficult to profit from."[54] Two economic characteristics of knowledge as a salable product are responsible for this deficiency: indivisibility of a

---

51. *See supra* notes 47-48.

52. Note, *supra* note 44, at 277; *accord* Bender, *Computer Programs: Should They Be Patentable?*, 68 Colum. L. Rev. 241, 244-45 (1968); Gemignani, *supra* note 2, at 292; *see also* amicus curiae brief of ADAPSO at 44, Parker v. Flook, 437 U.S. 584 (1978).

53. *See* R. Miller, *supra* note 18, at 304; J. Parker, *supra* note 4, at 294.

54. W. Nordhaus, Invention, Growth and Welfare: A Theoretical Treatment of Technological Change 70 (1969) (stating that the "provision of informational services is both an inappropriable investment and an increasing returns activity.").

software program from its novel features,[55] and appropriability of these features.

The indivisibility problem arises when software producers market their newly patented software innovations, because the sale of these innovations results in public inspection of their novel aspects. These novel aspects make software innovations attractive to consumers and allow the creators to demand high prices for these goods. The ability of software producers to charge high prices, however, is essential to their economic success because software developmental costs often are exorbitant.[56] Since software developers presently do not enjoy legally recognized property rights in their innovations, competitors are free to appropriate the novel aspects of these products when they appear for sale in the market.[57]

Once a rival has appraised the novel aspects of a competitor's innovation, it is free to reproduce the invention[58] and appropriate the fruits of the competitor's creative efforts. An appropriating rival often can undercut the innovator's selling price[59] because the innovator usually has incurred high fixed costs to formulate and develop new ideas into a marketable invention[60] while appropriation may cost only a few pennies.[61] This economic phenomenon inevitably attenuates the incentive to devote resources to the production of new knowledge such as software programs. Patent protection of software programs would curb the problem of appropriation by giving a program's creator a property right in the program.[62]

---

55. J. PARKER, *supra* note 4, at 294.

56. *See* Demsetz, *Information and Efficiency: Another Viewpoint*, 12 J.L. & ECON. 1, 12 (1969). "[W]e expect a free enterprise economy to underinvest in invention and research . . . because it is risky, because the product can be appropriated only to a limited extent [to the inventor himself], and because of increasing returns in use." K. Arrow, *Economic Welfare and the Allocation of Resources for Invention* in THE RATE AND DIRECTION OF INVENTIVE ACTIVITY 609, 619 (1962).

57. *Id.*

58. J. PARKER, *supra* note 4, at 294.

59. The producer who invents with the intent to recover developmental costs must balance the potential costs of development against the probability of recovering those costs. The factors weighing in this decision include the probable demand for the innovation, the likelihood that competitors will appropriate the innovation, and the ability to prevent competitors from entering its product market and absorbing its profits. *Cf.* J. PARKER, *supra* note 4, at 306.

60. F. SCHERER, *supra* note 3, at 444.

61. Root, *supra* note 27, at 209.

62. "One effect of [the patent system] is to create property rights in knowledge." P. JOHNSON, *supra* note 3, at 42.

B. Barriers to Entry

Software inventors somehow must recover developmental costs, and patent protection, if available, would help meet this need by giving inventors the right to exclude others from reproducing their patented programs. The software inventor's patent rights would protect against the entry of rival producers into the competition with the inventor concerning the use of its software program and would give the patent holder power to demand monopoly profits for the program.[63] These monopoly profits would recompense the inventor for its developmental investment.

Without a patent, however, software innovators must find other means to prevent rivals from appropriating their inventions.[64] They could attempt to prevent appropriation by concealing the novel dimensions of their software innovations or by relying upon their market power as leverage against appropriating companies. Most software producers, however, do not enjoy sufficient market power to persuade competitors not to appropriate their software innovations.[65] Concealing the novel aspects of software innovations is an equally ineffective and undesirable method of preventing appropriation because it inevitably causes software innovators to hoard their inventions, and acts as an impediment to technological advancements and free accessibility of knowledge.[66] In addition, this concealment method probably would prevent software producers from maximizing profits from the sale of their innovations because consumers presumably will not pay the maximum price for a product unless they can appreciate fully its valuable qualities. Thus, without patent protection, most software producers probably could not prevent competitors from appropriating their inventions.

C. Incentive and the Innovative Response

The availability of patent protection for inventions such as software provides inventors with tremendous incentive to devote resources to technological change.[67] Patent protection eliminates

---

63. W. NORDHAUS, *supra* note 54, at 60-64. F. SCHERER, *supra* note 3, at 442.

64. J. PARKER, *supra* note 4, at 301.

65. *See supra* note 42.

66. *See* Dunham, *The Necessity of Publishing Programs*, 25 COMPUTER J. 61 (1982).

67. "Only in the case of a patented product is a firm able to make the expenditures necessary to bring the advantages of the product to the attention of the customer without fear of competitive appropriation if the product proves successful." Kitch, *The Nature and Function of the Patent System*, 20 J.L. & ECON. 265, 277 (1977).

many of the commercial risks attendant to the marketing of knowledge as a product, such as the high probability of appropriation and low probability of profit, by granting the inventor exclusive monopoly rights in the product. Monopoly rights negate the fear of poor returns on technological investment by undercutting the ability of rival producers to use the patented technology to compete with the patent holder.[68] The patent holder, therefore, enjoys an improved chance of profiting from its ingenuity.

The grant of patent protection requires the inventor to disclose the secrets of the innovation. Disclosure of innovative secrets accelerates the diffusion of new ideas[69] and contributes to the achievement of optimal resource utilization by reducing unnecessary duplication of inventive efforts.[70] Widespread circulation of new ideas also increases the rate of technological development throughout an industry.[71]

D. The Expectational Effect

The availability of patent protection affects an inventor's decision to commit time and money to innovative effort by giving inventors an expectational incentive to undertake research and development.[72] The potential of acquiring a patent reward complements free market mechanisms by encouraging research and development in high cost ventures. The expectational incentive provides the impetus for initiation of high cost inventive efforts[73] and for continuation of projects when unforeseen costs arise. Similarly, the expectation of receiving a patent will encourage companies to engage in much more risky, but potentially rewarding, research and development programs.[74]

IV. PATENT LAW PRINCIPLES

The Patent Office and courts have denied applications for process patents on computer software programs, reasoning that

---

68. J. PARKER, *supra* note 4, at 302.

69. *Id.*

70. "[The] patent system enables firms to signal each other, thus reducing the amount of duplicative investment in innovation." Kitch, *supra* note 67, at 278. Patent holders have a strong incentive to inform many companies about their newly patented innovations in order to license the patent rights at a competitive price. *Id.*

71. J. PARKER, *supra* note 4, at 304.

72. J. PARKER, *supra* note 4, at 306. *See* F. SCHERER, *supra* note 3, at 446.

73. J. PARKER, *supra* note 4, at 306.

74. F. SCHERER, *supra* note 3, at 446.

software algorithms are not appropriate subject matter for patent protection.[75] This reasoning, however, is neither consonant with a clear understanding of software technology nor sensitive to the policy and goals of the patent system.

The Constitution grants Congress the power "[t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[76] Pursuant to this authority, Congress, by passing patent laws, has established criteria that limit the extension of patent rights only to inventions which demonstrate a potential for contributing a net benefit to society.[77] The Patent Act of 1952,[78] which is the most recent and comprehensive patent legislation that Congress has passed,[79] requires claimed inventions to fall within the category of appropriate patentable subject matter and to meet requirements of utility, novelty, and nonobviousness.[80] The Patent Act's requirements represent Congress' attempt to balance the policy of encouraging inventors to develop inventions that benefit society with the need for well defined patent issuance guidelines which promote orderly administration of the patent system.

A. The Subject Matter Requirement

Section 101 of the Patent Act describes the range of subject matter that is eligible for patent protection. It states that anyone who "invents or discovers any new and useful process, machine, manufacture, or composition of matter" may obtain a patent for

---

75. *See infra* notes 98-156 and accompanying text.

76. U.S. CONST. art. I, § 8, cl. 8.

77. The patent law requirement that a patent applicant demonstrate that a claimed invention potentially will benefit society is consistent with the constitutional goal of "promot[ing] the Progress of Science." *Id.* Patents, however, are legal monopolies and consequently are an exception to American society's general aversion to any form of unregulated monopoly power. For example, Thomas Jefferson, who strongly opposed monopolies in general, supported the creation of legal monopolies through patent grants and drafted the first Patent Act. He gave the following reasons for his view: "Certainly an inventor ought to be allowed the right to the benefit of his invention for some certain time. . . . Nobody wishes more than I do that ingenuity should receive liberal encouragement." 5 WRITINGS OF THOMAS JEFFERSON 75-76 (H. Washington ed. 1807). *See* F. MACHLUP, *supra* note 18, at 27-33.

78. 35 U.S.C. §§ 1-293 (1952) (current version at 35 U.S.C. §§ 1-376 (1976)).

79. For a discussion of the 1952 Patent Act and the changes made thereby, see Graham v. John Deere Co., 383 U.S. 1 (1965).

80. 35 U.S.C. §§ 101-03 (1976). For an excellent discussion of the Patent Act's requirements of novelty, utility, and nonobviousness, see P. ROSENBERG, PATENT LAW FUNDAMENTALS §§ 7.01-9.05 (2d ed. 1980).

this innovation.[81] The legislative history of section 101 demonstrates that Congress intentionally worded the section broadly so it would encompass wide areas of unforeseen invention.[82] The 1952 Patent Act Committee, for example, stated that section 101 encompasses "anything under the sun that is made by man."[83] Despite the breadth of section 101's subject matter eligibility language, administrative bodies and courts have interpreted this language strictly to insure that grants of patent protection are consistent with the basic policies of the patent system.[84]

B. Process Patents and the "Laws of Nature" Limitation

Section 101 specifically allows for process patents. Controversy, however, exists over the meaning and scope of the term "process."[85] In *Cochrane v. Deener*[86] the Supreme Court announced the following definition of process that courts have accepted for more than 100 years:

> A process is a mode of treatment of certain materials to produce a given result. It is an act, or a series of acts, performed upon the subject matter to be transformed and reduced to a different state of thing. . . . The process requires that certain things should be done with certain substances, and in certain order, but the tools to be used in doing this may be of secondary consequence.[87]

Thus, a process is patentable if it operates upon " 'certain materials' to 'produce a given result.' "[88] Application of this ambiguous definition has caused courts to identify some subject matters that do not qualify for patent protection, particularly ideas,[89] mental processes,[90] and discoveries of scientific principles or laws of nature.[91]

---

81. 35 U.S.C. § 101 (1976).

82. S. REP. No. 1979, 82d Cong., 2d Sess. 5 (1952), *reprinted in* 1952 U.S. CODE CONG. & AD. NEWS 2394, 2399; H.R. REP. No. 1923, 82d Cong., 2d Sess. 6 (1952).

83. *Id.*; *see In re* Prater, 415 F.2d 1393, 1406 (C.C.P.A. 1969) (Worly, C.J., concurring).

84. *See* Graham v. John Deere Co., 383 U.S. 1 (1965) and authorities cited therein.

85. Note, *supra* note 15, at 714.

86. 94 U.S. 780 (1876) (patent granted for the development of an improved process of grinding flour).

87. *Id.* at 788. The definition section of the Patent Act states that " 'process' means process, art or method, and includes a new use of a known process, machine, manufacture, composition of matter, or material." 35 U.S.C. § 100(a) (1976).

88. Novick & Wallenstein, *supra* note 12, at 316 (quoting Cochrane v. Deener, 94 U.S. 780 (1876)).

89. Gottschalk v. Benson, 409 U.S. 63, 67 (1972); Rubber-Tip Pencil Co. v. Howard, 87 U.S. 498, 507 (1874).

90. Gottschalk v. Benson, 409 U.S. at 67-68.

91. P. ROSENBERG, *supra* note 80, at § 1.04.

This group of unpatentable subject matters is distinguishable from patentable processes. Discoveries of scientific principles or laws of nature, for example, are not patentable because they do not produce something that previously did not exist.[92] Rather discoveries such as Newton's formulation of the law of gravity merely identify something that existed but which mankind had failed to recognize prior to discovery.[93] The Supreme Court has held that discovery of pre-existing scientific principles or natural laws does not give the discoverer "the right to exclude others from enjoying the benefits derived from the operation of" the laws or principles.[94] Patentable processes, on the other hand, although they may require application of unpatentable natural laws, basically consist of an act or operation, or a series of acts or operations, performed upon specific subject matter to produce a physical result that is new and useful to society.[95]

The Patent Office and courts consistently have recognized "that applications of natural phenomena or scientific principles are patentable" if the patent does not preempt society from enjoying the benefits derived from the underlying principle.[96] The determination of whether granting patent protection to a certain invention would effectively give the inventor a monopoly over use of the un-

---

92. *Id.*

93. *Id.* Peter D. Rosenberg, a Primary Examiner at the United States Patent & Trademark Office, distinguished between a patentable invention and an unpatentable discovery in the following example:

> A classic illustration of the distinction between the discovery of a scientific principle or law of nature and an invention and the relationship between the two is furnished by the contributions of Benjamin Franklin on the subject of lightning. Franklin's recognition of the electrical nature of lightning falls into the category of a *discovery*, whereas his application of the discovery in the form of the lightning rod is an invention. While a scientific truth, or the mathematical expression of it, is not patentable invention, a novel and useful structure created with the aid of knowledge of scientific truth may be.

Id. at 1-16 (quoting Mackay Radio & Tel. Co. v. Radio Corp., 306 U.S. 86, 94 (1939)) (emphasis in original).

94. P. ROSENBERG, *supra* note 80, at § 1.04. Mr. Rosenberg stated that:

> It would, indeed, be a vain and foolish gesture to grant a mortal the right to exclude others from enjoying the benefits derived from the operation of [a natural] law. In attempting to assert that right, one would have to do, in effect, what King Canute did—command the tide not to come in!

Id.

95. *Id.* at § 6.01[1].

96. Comment, Diamond v. Diehr: *The Patentability of Processes and Incorporated Algorithms*, 8 OHIO N.U.L. REV. 535, 541 (1981) (citing Parker v. Flook, 437 U.S. 584, 590 (1978)); Blumenthal & Riter, *Statutory or Non-Statutory?: An Analysis of the Patentability of Computer Related Inventions*, 62 J. PAT. OFF. SOC'Y 454, 455-56 (1980)).

derlying principle, however, is a difficult analytical problem.[97] This problem becomes paramount when the Patent Office and courts face requests for patents covering new technological advances. Courts engage in a cost-benefit analysis when deciding whether to extend patent protection to a new technology. In this analysis, courts weigh the cost of granting an inventor exclusive rights to the invention's use against the benefit society receives from patent law encouragement of scientific innovation and advancement.

V. Supreme Court Treatment of Software Patent Cases

A. *Supreme Court Software Patent Cases*

1. 1972: *Gottschalk v. Benson*

In *Gottschalk v. Benson*[98] the Supreme Court considered for the first time whether computer software deserved patent protection. In *Benson* Gary Benson and Arthur Tabbot sought to patent "a method of programming a general-purpose digital computer to convert signals from binary-coded form to pure binary form" (the "Benson Program").[99] The Patent and Trademark Office (PTO) rejected their patent request, reasoning that the Benson Program was a set of unpatentable mental processes and mathematical steps.[100] The Court of Customs and Patent Appeals (CCPA) reversed[101] the PTO's decision and applied the "technological or useful arts" doctrine[102] to hold that the Benson Program satisfied the

---

97. One court stated that "[p]atents . . . approach, nearer than any other class of cases . . . to what may be called the metaphysics of the law, where distinctions are . . . very subtle and refined, and, sometimes, almost evanescent." Folsom v. Marsh, 9 F. Cas. 342, 344 (C.C.D. Mass. 1841) (No. 4,901).

98. 409 U.S. 63 (1972).

99. *Id.* at 65. A binary-coded decimal numeral (BCD) merely is a representation of each decimal digit in a multi-digit number with a binary-coded decimal number. "Thus decimal 53 is represented as 0101 0011 in BCD, because decimal 5 is equal to binary 0101 and decimal 3 is equivalent to binary 0011. In pure binary notation, however, decimal 53 equals binary 110101." *Id.* at 67.

100. *In re* Benson, 441 F.2d 682, 686-88 (C.C.P.A. 1971). The mental steps doctrine is a patent law doctrine which states that ideas are not patentable. Novick & Wallenstein, *supra* note 12, at 317 (footnotes omitted). Thus, a "process that consists entirely of mental steps is not patentable subject matter because it would create a monopoly over a law of nature or a disembodied idea." *Id. See* Halliburton Oil Well Cementing Co. v. Walker, 146 F.2d 817 (9th Cir. 1944); *In re* Shao Wen Yuan, 188 F.2d 377 (C.C.P.A. 1951); Davis, *supra* note 2, at 8.

101. *In re* Benson, 441 F.2d at 688.

102. Nimtz, *supra* note 47, at 13. Note, *supra* note 15, at 716. The CCPA developed the technological arts test in response to the PTO's practice of using the mental steps doctrine to deny software claims. Under the technological arts test, the CCPA presumes patent applicants have established a prima facie case that the subject matter of the inventor's ap-

standards of patentability under section 101 of the Patent Act because it furthered the useful arts.

The Supreme Court reversed the CCPA, denied the patent requests and concluded that the Benson Program was not a patentable process under section 101 of the Patent Act.[103] After reviewing process patent precedent, the Court stated that a process is patentable only if it is "tied to a particular machine or apparatus or [operates] to change articles or materials to a 'different state or thing.' "[104] The Court equated the Benson Program for binary-coded decimal conversion with the underlying unpatentable mathematical formula for carrying out these calculations.[105] It then concluded that the patent request for the Benson Program was an attempt to patent a natural law, which if allowed, would preempt society from using the underlying mathematical formula without permission from the Benson Program's creators.[106]

This portion of the Court's *Benson* opinion suggested that other software programs might meet its definition of a patentable process, even if the programs utilized unpatentable natural laws, if they instructed computer hardware devices to perform novel and useful processes.[107] The Court, however, then referred to a report of the President's Commission on the Patent System[108] which stated that, largely for administrative reasons, software programs should not be patentable.[109] The Court concluded that Congress,

---

plication is technical and not merely mental when the patent application discloses a technical apparatus that implements a process. Nimtz, *supra* note 47, at 12-13.

103. Gottschalk v. Benson, 409 U.S. 63, 71-73 (1972).

104. *Id.* at 71 (citing Smith v. Snow, 294 U.S. 1, 19-20 (1935) (patent granted for a method of setting eggs in a staged incubator and circulating warm air); Expanded Metal Co. v. Bradford, 214 U.S. 366, 385-86 (1909) (process patent granted for a method of expanding metal); Tilghman v. Proctor, 102 U.S. 707, 729 (1880) (patent granted for a process of "manufacturing fat acids and glycerine from fatty bodies by the action of water at high temperature and pressure")).

105. Gottschalk v. Benson, 409 U.S. at 71-72.

106. *Id.*

107. *See* Gottschalk v. Benson, 409 U.S. at 71-72.

108. *Id.* at 72 (citing REPORT OF THE PRESIDENT'S COMM'N ON THE PATENT SYS., TO PROMOTE THE PROGRESS OF . . . USEFUL ARTS IN AN AGE OF EXPLODING TECHNOLOGY (1966)).

109. The report of the President's Commission on the Patent System provides in pertinent part:

> The Patent Office now cannot examine applications for programs because of a lack of a classification technique and the requisite search files. Even if these were available, reliable searches would not be feasible or economic because of the tremendous volume of prior art being generated. Without this search, the patenting of programs would be tantamount to mere registration and the presumption of validity would be all but nonexistent.

Id.

with its broader investigatory power, should be the governmental body that ultimately decides whether software programs receive patent protection.[110] This abdication of decision-making authority to Congress suggests that the Court would not grant patent protection to software programs in the future.

2. 1978: *Parker v. Flook*

In *Parker v. Flook*[111] the Supreme Court reaffirmed *Benson* and strongly suggested that it would not extend patent protection to computer software in the future. The plaintiff in *Flook* applied for a patent on a method for updating alarm limits on process variables during the catalytic conversion of hydrocarbons.[112] In catalytic conversion processes, when any process variable such as temperature, pressure, and flow rate "exceeds a predetermined 'alarm limit,' an alarm may signal the presence of an abnormal condition indicating either inefficiency or perhaps danger."[113] Flook created a software program (the "Flook Program") that instructed a computer to measure periodically the process variables and to use a previously undiscovered mathematical formula to update the alarm value.[114] Although fixed alarm limits are adequate when the process variables are steady, production technicians must update alarm limits during transient periods in the catalytic conversion when process variables are changing.[115] The Flook Program provided a new method of updating alarm limits during these transient periods.

The PTO rejected Flook's patent application by reasoning that the new mathematical formula constituted the only difference between Flook's claims and prior art.[116] Thus, the PTO concluded that a patent on the Flook Program realistically would have been a patent on Flook's unpatentable mathematical formula.[117] The PTO Board of Appeals sustained the PTO's decision because the " 'point of novelty in [Flook's] claimed method' lay in the formula or algorithm described in the claims, a subject matter that was un-

110. Gottschalk v. Benson, 409 U.S. at 73.
111. 437 U.S. 584 (1978).
112. *Id.* at 585.
113. *Id.*
114. *Id.* at 585-86.
115. *Id.*
116. *Id.* at 587.
117. *Id.*

patentable."[118] The CCPA, however, reversed the Board's decision.[119] It said that *Benson* applied only to claims that entirely preempted a mathematical formula or algorithm and noted that Flook sought to patent his program only as it applied to his modification of a scientific process.[120] Thus, the CCPA granted Flook's patent request by reasoning that it was not an attempt to patent a mathematical principle.[121]

The Supreme Court assumed that Flook's mathematical formula was the only novel portion of Flook's process and reversed the CCPA. It reasoned that Flook's formula was an unpatentable discovery.[122] The Court also stated that merely using the figures computed according to Flook's newly discovered formula to adjust alarm limits did not make his process patentable.[123] The Court emphasized that its decision should not "be interpreted as reflecting a judgment that patent protection of certain novel and useful computer programs [would] not promote the progress of science and the useful arts, or that such protection is undesirable as a matter of policy."[124] As in *Benson*,[125] the Court remarked that Congress was the appropriate tribunal for deciding "[d]ifficult questions of policy concerning the kinds of programs that may be appropriate for patent protection and the form and duration of such protection . . . ."[126]

Justice Stewart, writing for the dissent,[127] distinguished Flook's patent request from the patent request in *Benson* and agreed with the CCPA that Flook's process satisfied the standards of subject matter patentability under section 101 of the Patent Act.[128] He stated that the Court in *Benson* held claims for a software algorithm unpatentable, which if granted, would have given the algorithm's creators a monopoly over an unpatentable natural law.[129] Justice Stewart argued that Flook, in contrast,

---

118. *Id.*
119. *Id.*
120. *Id.*
121. *In re* Flook, 559 F.2d 21, 23 (C.C.P.A. 1977).
122. Parker v. Flook, 437 U.S. at 589-90.
123. *Id.* at 590.
124. *Id.* at 595.
125. See *supra* notes 98-110 and accompanying text for a discussion of Gottschalk v. Beuson.
126. Parker v. Flook, 437 U.S. at 595.
127. Chief Justice Burger and Justice Rehnquist joined Justice Stewart in dissent. *Id.* at 598-600.
128. *Id.* at 600.
129. *Id.* at 599.

sought to patent an improved process for calculating alarm limits for process variables in catalytic conversion procedures.[130] Moreover, he emphasized that Flook's process should be patentable even though one step contained subject matter that would not have been patentable by itself.[131] He assumed "that thousands of processes and combinations have been patented that contained one or more steps or elements that themselves would have been unpatentable subject matter."[132] Thus, the Justice concluded that Flook's process met the Patent Act's subject matter patentability standards and that the majority improperly imported the tests of novelty and inventiveness of sections 102 and 103 to deny Flook's claim.[133]

3. 1981: *Diamond v. Diehr*

In *Diamond v. Diehr*[134] the Supreme Court granted respondents' application for a patent on a process they devised for molding raw, uncured synthetic rubber into cured precision products.[135]

---

130. *Id.*

131. *Id.*

132. *Id.* at 599-600.

133. *Id.* at 600. One commentator also criticized the *Flook* majority opinion for not complying with a fundamental principle of patent claim analysis. *See* Note, *The Patentability of Computer Software*, 1979 WIS. L. REV. 867, 886-87. Section 103 of the Patent Act requires courts to evaluate patent claims as a whole rather than in fragments. This commentator argued that the *Flook* majority ignored § 103's mandate by first isolating Flook's mathematical formula from the rest of his patent claim and assuming that it was prior art. *Id.* at 886. Thus, the Court explicity removed Flook's mathematical formula from its consideration of his entire patent claim. *Id.* The Court then examined the remainder of Flook's application "as a whole." *Id.*

The CCPA decisions following *Flook* are divisible into three general groups. *See* Note, *supra* note 44, at 295. First, when a software claim does not include an algorithm, the CCPA has held that the claimed invention satisfied § 101's subject matter requirements. *See, e.g., In re* Bradley, 600 F.2d 807 (C.C.P.A. 1979), *aff'd sub nom. Diamond v. Bradley*, 450 U.S. 381 (1981). Second, when a software claim includes an algorithm but also enumerates other elements as the source of a claimed invention's innovation, the CCPA has held it to be patentable subject matter. *See, e.g., In re* Diehr, 602 F.2d 982 (C.C.P.A. 1979), *aff'd sub nom.* Diamond v. Diehr, 450 U.S. 175 (1981); *In re* Johnson, 589 F.2d 1070 (C.C.P.A. 1978), *rev'd on other grounds sub nom.*, Dann v. Johnston, 425 U.S. 219 (1976). Last, when a software claim is purely mathematical, the CCPA has held them to be unpatentable subject matter. *See, e.g., In re* Maucorps, 609 F.2d 481 (C.C.P.A. 1979); *In re* Gelnovateh, 595 F.2d 32 (C.C.P.A. 1979).

134. 450 U.S. 175 (1981).

135. *Id.* at 177. Before respondents in *Diehr* invented their rubber curing method, the rubber industry had been unable to develop a process for determining proper curing times largely because it could not obtain precise temperature measurements inside curing presses. *Id.* at 178 & n.4. Respondents devised a process that continuously measured the temperature inside the press and automatically relayed temperature data to a computer that repeatedly recalculated the remaining time for curing. *Id.* at 178. Respondents' process used the

The Court endorsed the claimed innovation as an improved industrial process even though it required the use of a digital computer.[136] Despite its approval of the patent application, however, the Court again accepted the erroneous characterization of software algorithms that it developed in *Benson* and *Flook*.[137] Thus, *Diehr* suggests that although processes that use computers may be patentable, the Court probably will not grant patent protection to software programs themselves.

The patent examiner in *Diehr* relied upon *Benson* and determined that respondents' use of a computer in its rubber curing process constituted non-statutory subject matter.[138] The examiner then concluded that the remaining steps were conventional unpatentable steps in the rubber curing process.[139] The PTO Board of Appeals agreed with the examiner, but the CCPA reversed by reasoning that a process that satisfies the subject matter requirements of section 101 of the Patent Act does not become unpatentable simply because it calls for the use of a computer.[140] The CCPA additionally remarked that the respondents' claim was not an attempt to patent a mathematical algorithm and characterized it as the "resolution of a practical problem [that] had arisen in the molding of rubber products."[141]

The Supreme Court affirmed the CCPA's decision and held that respondents' claim recited subject matter that was eligible for patent protection. Justice Rehnquist, writing for the majority, characterized the Diehr claim as an improved industrial process, not an attempt to patent a mathematical formula.[142] He stated, however, that even if a patent claim for a process contains a mathematical formula, it will satisfy the subject matter requirements of section 101 if the process, when considered as a whole, implements or applies the formula to perform "a function which the patent laws were designed to protect (e.g., transforming or reducing an article to a different state or thing) . . . ."[143]

Justice Stevens, writing for the dissent, rejected the majority's

---

Arrhenius equation to calculate cure times in rubber molding presses. *See id.* at 177 n.2 for a discussion of the Arrhenius equation.

136. *Id.* at 185-91.
137. *See supra* notes 98-133 and accompanying text.
138. Diamond v. Diehr, 450 U.S. at 180.
139. *Id.* at 181.
140. *Id.*
141. *Id.*
142. *Id.* at 187, 191-93.
143. *Id.* at 192.

holding that Diehr's patent claim presented a new, patentable process for curing synthetic rubber.[144] He argued that the claimed invention in *Diehr*, like that in *Flook*,[145] "was an algorithm that could be programmed on a digital computer."[146] He found that the most significant distinction between the *Diehr* and *Flook* inventions was in the drafting of the patent claims, not in the inventions themselves.[147] Justice Stevens, therefore, stated that the majority should have applied a *Flook* analysis to decide the *Diehr* claim.[148] Justice Stevens emphasized that, under the *Flook* analysis, the majority should have separated the *Diehr* algorithm from the rest of the *Diehr* claim, characterized the algorithm as an unpatentable law of nature under section 101 of the Patent Act, and then searched the remaining part of the patent claim for patentable subject matter.[149] Under this analysis, Justice Stevens argued that the *Diehr* process, with the exception of the unpatentable computerized calculation of curing times, merely consisted of well known conventional methods of curing rubber.[150]

Some commentators argue that *Diehr* foreshadows a judicial trend toward granting patent protection to computer software programs.[151] One commentator predicted that *Diehr* would "cause a revolution in the practice of the Patent and Trademark Office" and that "applications for inventions embodied in computer software will be examined, allowed and issued as patents . . . ."[152] Unfortunately, the *Diehr* opinion does not support this overly optimistic view. *Diehr* seems to stand only for the narrower proposition that courts may not declare a process that satisfies the subject matter requirements of section 101 for process patents ineligible for patent protection simply because some steps in the process call for the use of a mathematical equation or a programmed digital computer.[153] Under this approach, however, the Patent Office will evaluate process patent claims by treating anything that relates to

144. *Id.* at 219-20.

145. *Id.* at 209-10.

146. *Id.* at 209.

147. *Id.* at 210 n.32 (citing Blumenthal & Riter, *supra* note 96, at 502-03).

148. Diamond v. Diehr, 450 U.S. at 212-14.

149. *Id.*

150. *Id.* at 208-09, 212-14.

151. *See e.g.*, Blumenthal, *Supreme Court Sets Guidelines for Patentability of Computer Related Inventions*—Diamond v. Diehr, 63 J. PAT. OFF. SOC'Y 117 (1981); Nimtz, Diamond v. Diehr: *A Turning Point*, 8 RUT. J. COMPUTERS, TECH. AND THE LAW 267 (1981).

152. *See* Nimtz, *supra* note 151, at 267.

153. 450 U.S. at 185.

a computer or its software as an unpatentable scientific truth and separately evaluating the patentability of the remaining portion of the process. Thus *Diehr* authorizes patent protection only for processes that historically would have been patentable despite the use of a computer in the process.

Furthermore, *Diehr* reinforced the Court's erroneous characterization of software algorithms.[154] The *Diehr* majority re-emphasized the *Benson* definition of an algorithm as "a 'procedure for solving a given type of mathematical problem,' " and went on to say that "an algorithm . . . cannot be the subject of a patent."[155] The Court evidently does not understand that by granting the Diehr process patent protection it authorized patent protection of the software algorithm that controlled the operation of Diehr's process. The algorithm in the Diehr process was not just the mathematical equation used to express the temperature and pressure relationship necessary for curing synthetic rubber. Rather, the Diehr program, and its underlying algorithms, completely controlled and made possible the process that solved the problem of obtaining accurate cure times for synthetic rubber.[156] Thus, the Court's continued mischaracterization in *Diehr* of software algorithms as unpatentable natural laws remains a major reason why the Court will not extend patent protection to software programs in the near future.

B. Supreme Court Mischaracterization of Software Algorithms

The Supreme Court has defined algorithms as procedures "for solving a given type of mathematical problem"[157] and has characterized them as unpatentable natural laws.[158] The Court's conceptualization of algorithms is erroneous for several reasons. First, the Court improperly limited its definition of algorithms to procedures that solve mathematical problems.[159] Problem solvers have not confined their use of algorithms to mathematical contexts.[160] Rather, they have used algorithmic processes to solve many differ-

---

154. *See infra* notes 157-74 and accompanying text.

155. 450 U.S. at 186.

156. *See supra* note 31 and accompanying text.

157. Parker v. Flook, 437 U.S. at 585 n.1 (quoting Gottschalk v. Benson, 409 U.S. at 65).

158. Parker v. Flook, 437 U.S. at 589 (in which the Court stated that "an algorithm, or mathematical formula, is like a law of nature . . . .").

159. Diamond v. Diehr, 450 U.S. at 186.

160. *See* I. HORABIN & B. LEWIS, *supra* note 31, at 8-18.

ent problems and to perform a variety of tasks.[161]

Second, not all computer software algorithms are equivalent to unpatentable natural laws. Rather, courts should view software algorithms, like other scientific processes,[162] as lying on a continuum between clearly patentable and clearly unpatentable processes. Software algorithms should lie within an unpatentable portion of this continuum if they relate very closely to a preexisting natural law.[163] For example, a software algorithm that mechanically calculates the length of a right triangle's hypotenuse by using the Pythagorean theorem[164] would not be patentable. If it was patentable, the program's creator effectively would enjoy a patent monopoly over the Pythagorean theorem, a natural law.[165] A software algorithm, however, should lie within the patentable portion of this continuum, even if its bases are natural laws, if it directs a computer to perform a new and useful process.[166] Merrill Lynch, Pierce, Fenner & Smith currently uses a software program that has received patent protection.[167] This program controls a management account system (the "CMA") that regulates the flow and management of money invested in an integrated cash management program comprised of a mutual fund, a checking account, and credit card account.[168] Although mutual funds, checking accounts, and credit card accounts all were available before Merrill Lynch introduced its CMA, the system provides investors with new "synergistic benefits" that are not obtainable by investing separately in each type of account.[169] This program deserves patent protection be-

---

161. *Id. See supra* notes 31-37 and accompanying text.

162. *See e.g.*, Smith v. Snow, 294 U.S. 1 (1935); Waxham v. Smith, 294 U.S. 20 (1935) (holding as patentable a process for setting eggs in staged incubation and applying mechanically circulated currents of air to eggs to ensure proper incubation temperatures).

163. *See* Parker v. Flook, 437 U.S. at 589.

164. See Novick & Wallenstein, *supra* note 12, at 336, for a discussion of the Pythagorean theorem.

165. *See* Parker v. Flook, 437 U.S. at 589.

166. *See* 35 U.S.C. § 101 (1954).

167. Paine, Webber, Jackson & Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith, Inc., 564 F. Supp. 1358 (D. Del. 1983).

168. *Id.* at 1363-66.

169. Merrill Lynch's patent described only the algorithmic flow chart for its CMA. It did not describe any of the mechanical apparatus that operation of the CMA required. *Id.* at 1367. The court held that the patent protected the software algorithm and used the reasoning that this Note advocates. *Id.* at 1372-73. The CMA provides customers with the following new "synergistic benefits":

> [First,] all money generated in the Securities Account is automatically invested within a week into the Money Market Fund. This differs from a conventional brokerage account, which might not invest money generated from activity in the brokerage account

cause it used natural laws to create a new and useful money management and investment system.

Last, the Court equated software algorithms with unpatentable mathematical formulas without realizing that these algorithmic processes satisfy its interpretation of the Patent Act's "process" definition.[170] In *Parker v. Flook*[171] the Court stated that it arguably has recognized a process as falling within the Patent Act's definition only when "it either was tied to a particular apparatus or operated to change materials to a 'different state or thing.' "[172] Some algorithms, of course, like those that solve complex mathematical problems which require only a mathematician's mental powers to be completely useful,[173] would not fall within the scope of this definition because they are usable without the aid of a "particular apparatus."[174] Software algorithms, however, interact

---

and thus some money might remain in an account without yielding any financial return. These proceeds, referred to as "idle cash" do not enhance the customer's portfolio and usually is not compatible with the customer's overall financial objectives. By investing any idle cash generated in the Securities Account, into the Money Market Fund, the customer apparently receives a greater return on his initial investment and therefore is consistent with the customer's overall objectives.

. . . .

[Second,] the cash balances in the Securities Account, shares in the Money Market Fund, and available margin loan value of the securities in the Securities Account are calculated when determining the amount of credit available in the Visa Account. Also, payments made by Merrill Lynch to Bank One in payment of Visa balances, on behalf of the CMA customers, are made in the following order of priority: (1) from the cash balances, if any held in the Securities Account; (2) from the proceeds of redemption of Money Fund shares in CMA accounts; and (3) from margin loans to the customer by Merrill Lynch within the available margin loan value of the securities in the Securities Account. This system of priority arguably provides for an efficient use of funds because the customer will not incur the cost of a margin loan until all free credit cash balances and funds invested in Money Market Fund shares are fully utilized.

. . . .

[Last,] those customers who subscribe to the CMA receive a monthly transaction statement from Merrill Lynch which details all CMA transactions during the preceding month. The statement describes securities and options bought and sold in the Securities Account, whether on margin or on a fully-paid basis, any other type of transaction effected in the Securities Account, margin interest charges, if any, Money Market Fund shares that were purchased or redeemed, dividends on Money Market Fund shares, purchases of merchandise or services that were made with the Visa card, checks drawn against the Visa Account and cash advances.

Id. at 1361-62.

170. Parker v. Flook, 437 U.S. at 588 n.9; 35 U.S.C. § 101 (1954).

171. 437 U.S. 584 (1978).

172. *Id.* at 588 n.9.

173. See Uspensky & Semenov, *supra* note 34, at 100-234, for a discussion of the use of algorithms in mathematics.

174. Parker v. Flook, 437 U.S. at 588 n.9.

inseparably with computer hardware devices to perform the processes that the algorithms specify. Computer software and hardware are useless without each other. Thus, software algorithms should fall within the scope of the Court's definition of patentable process.

VI. Analysis of Software Patentability: A Policy Question

The basic purpose of the patent system is to encourage the production and disclosure of new knowledge by offering inventors an opportunity to recover the costs of inventing and to profit from their inventions. The patent system achieves this purpose by granting inventors exclusive monopoly rights to make, use, or sell their innovation for a limited period of time. Society benefits from patent grants by receiving access to and use of the patented invention. Society pays for patent grants by allowing inventors to charge monopoly prices for use of their inventions and by protecting them from competition. Patents, therefore, should protect inventions only if the protection's benefit to society outweighs the costs it forces society to incur.

A. The Benefits of Granting Computer Software Patents

Society would benefit in several ways if patents protect software programs in the future. First, patent protection would cause an increase in technological knowledge in the software industry. Second, it would cause this new knowledge to spread throughout the industry with increased speed and trigger new software inventions. Third, new software innovation would help society use scarce resources more efficiently.[175] Last, patent protection would cause more and better applications for software products to arise and improve the quality of products software helps to produce.

If software programs receive patent protection, the disclosure requirements of the patent laws naturally would cause the stock of technological knowledge in the software industry to increase. The patent application process requires applicants to disclose and explain the most innovative aspects of their inventions.[176] Disclosure would make this information freely accessible to anyone who requests it from the Patent Office.[177] This disclosure requirement also fosters rapid dissemination of the novel aspects of new inven-

175. *See* Kitch, *supra* note 67, at 279.
176. *See* 35 U.S.C. §§ 112-15 (1954).
177. *See* J. Parker, *supra* note 4, at 303; 37 C.F.R. § 1.11 (1982).

tions within technological communities. The combination of an increase in the stock of technological knowledge and in the rate of knowledge dissemination in the software industry resulting from patent protection would fuel and reinforce the inventive process. Further, the availability of patents would give software innovators an expectation of patent protection and would encourage them to undertake more and riskier quests for new innovation.[178]

The widespread publication of new software innovations resulting from the patent application process also would prevent wasteful duplication of research and development efforts. More efficient utilization of inventive effort in the software field would promote the development of many new and better applications for software programs. This growth in the use of computers naturally would result in higher productivity throughout the economy.[179]

The technological advancement in the computer industry that patent protection of software programs would encourage probably would improve the general standard of living in America. Several empirical studies have concluded that technological advancement is a major reason for growth in per capita income in the Western world over the last few centuries.[180] In addition, if modern American society continues its trend of emphasizing the importance of information processing and data collection, the growth and health of America's economy increasingly will depend upon advances in computer software technology. Patent protection of computer software would help insure technological advancement in this area.

B. The Costs of Granting Computer Software Patents

Society will incur several costs if patents protect software programs in the future. First, software patents will give patent holders the power to charge consumers monopoly prices for the use and purchase of patented software. Second, patent holders sometimes misuse patent rights in an invention to the detriment of society. Last, the availability of software patents inevitably would cause software inventors to flood the Patent Office and courts with patent claims. These claims would lead to exorbitant administrative and judicial costs.

Software patents will force society to give patent holders mo-

178. *See supra* notes 72-74 and accompanying text.

179. *See* Ernst, SCIENTIFIC AMERICAN, *supra* note 1 at 144.

180. W. NORDHAUS, *supra* note 54, at 8; Barzel, *Optimal Timing of Innovations*, 50 REV. ECON. STAT. 348, 354 (1968).

nopoly pricing power over the use and sale of the patented inventions. In theory, monopoly power causes monopolists to offer consumers fewer goods at higher prices than they would in a competitive economy.[181] Software innovators inevitably must charge high prices for the use or sale of their product to recover the exorbitant research and development costs they incurred in creating the software. Additionally, the mere knowledge of monopoly power over a product would induce software patent holders to charge consumers higher prices for the software.

Two economic mechanisms would mitigate the monopoly power that software patents would confer. First, the price of old software programs that consumers readily could substitute for the new patented program limits the patent holder's monopoly pricing power. The sales of new patented software programs theoretically will fall if the patent holder prices them significantly above the prices of substitute software.[182] Second, if a newly patented product has no close substitutes and the patent holder charges monopoly prices for the product, then other inventors probably will seek to develop products that will not violate the patent, thereby legally capturing some of the patent holder's monopoly profits.

Some patent holders legally misuse patent rights in a manner that suppresses technological advancement in a certain scientific field. Patent misuse may occur, for example, when patent holders prevent society from using or benefiting from their invention.[183] It also may occur when an enterprise engages in an aggressive effort to patent all potential advances and improvements in a particular line of technology. When this effort is successful, the patent holder effectually has created a legal barrier which blocks competition in that technological area for the lives of the patents. Such patent misuse controverts the policies of the patent system and impedes technological progress.

Finally, because of the high administrative and judicial costs that accompany the prosecution and adjudication of patent claims, the Patent Office[184] and many commentators[185] have opposed the

---

181. Kamerschen, *The Economic Effects of Monopoly: A Lawyer's Guide to Antitrust Economics*, 27 MERCER L. REV. 1061, 1062 (1976).

182. *See* R. POSNER, ANTITRUST LAW: AN ECONOMIC PERSPECTIVE 126-27 (1976).

183. *See supra* note 7.

184. *See* Parker v. Flook, 437 U.S. at 587-88 (The acting Commissioner of Patents and Trademarks argues that patent protection of software algorithms "will have a debilitating effect on the rapidly expanding computer 'software' industry, and will require him to process thousands of additional patent applications.").

185. Gemignani, *supra* note 12, at 312; REPORT OF THE PRESIDENT'S COMM'N ON THE

issuance of software patents. While this problem is at odds with the Constitution's goal of furthering scientific achievement, society still must bear these administrative and judicial costs if software receives patent protection.

C. *Issuing Software Patents: Need for an Administrative Response*

The explosive growth of computer usage in most aspects of human activity evinces society's compelling need for the extension of patent protection to computer software innovations. Man's technological ingenuity seems to be the only limitation on the benefits[186] society would receive by encouraging software innovation through patent protection. Conversely, the costs of extending patent protection to computer software[187] represent a familiar set of considerations that the judiciary and patent administrators have addressed and overcome throughout the history of the patent system.[188]

The Supreme Court has acknowledged that one of its major reservations about granting patent protection to computer software is the tremendous administrative burden that this protection would force upon the Patent Office.[189] The Court noted that extending patent protection to computer software would require the

---

PATENT SYSTEM 13 (1966). *Contra* Note, *An Anomaly in the Uncertain Status of Computer Software*, 8 RUT. J. COMPUTERS, TECH. AND THE LAW 273, 277 (1981).

186. *See supra* notes 175-80 and accompanying text.

187. *See supra* notes 181-85 and accompanying text.

188. One commentator, for example, succinctly described the historic hut well-settled policy debate concerning the propriety of granting inventors monopoly power over their inventions as follows:

> While scientists, and even judges, have on occasion spoken disparagingly of the motives of those who would be so mercenary as to seek patents, the fact must not be overlooked that inventors, however superhuman they may appear in the popular mind, require and should be entitled to material sustenance at least at a level commensurate with less creative segments of society. The economic philosophy hehind the Constitutional clause empowering Congress to grant patents and copyrights is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors and inventors in "Science and useful Arts." Sacrificial days devoted to such creative activities deserve rewards commensurate with the services rendered. As was long ago recognized by the Supreme Court, absent a monopoly, an imitator would enjoy a substantial competitive advantage over the innovator in that the imitator, not having expended capital to create and develop the innovation, could afford to sell it at a lower price than the inventor.

P. ROSENBERG, *supra* note 80, at § 1.07.

189. Gottschalk v. Benson, 409 U.S. at 72 (quoting THE REPORT OF THE PRESIDENT'S COMM'N ON THE PATENT SYSTEM, TO PROMOTE THE PROGRESS OF . . . USEFUL ARTS IN AN AGE OF EXPLODING TECHNOLOGY 14 (1966)).

Patent Office to process a large number of new patent applications without the ability to compare these applications to the tremendous volume of previously created software programs.[190] Because the Court improperly has characterized software algorithms as unpatentable natural laws,[191] this administrative fear and the Court's feeling that Congress would be a better forum[192] for the resolution of this patentability question seem to be the only legitimate barriers to software patentability.

An addition to the Patent Office of a small staff of computer science experts probably would remedy the administrative problem of processing software patent applications. One computer science expert[193] has stated that reviewing software patent applications for novelty, usefulness, and nonobviousness[194] would not be difficult for most computer science experts. Many of these experts possess extensive knowledge of current software technology and closely monitor major software improvements.[195] Thus, the Patent Office probably could resolve the administrative burden of processing software patent applications without completely renovating its present operating structure. This small administrative change would bolster the patent system's effectiveness as a social tool for promoting scientific innovation and advancement.

VII. CONCLUSION

The patent system is a policy tool designed to achieve social welfare goals. The patent system rewards inventors who develop new scientific inventions by granting them a limited monopoly over their inventions. These rewards encourage inventors to invest their time and money in scientific inquiry. An inventor receives patent protection, however, only when an invention confers benefits to society that exceed the social costs of granting the inventor a patent monopoly.

Computer software programs have not received patent protection primarily because the United States Supreme Court improp-

---

190. *Id.*

191. *See supra* notes 157-74 and accompanying text.

192. Gottschalk v. Benson, 409 U.S. at 73; Parker v. Flook, 437 U.S. at 595-96.

193. Interview with Professor Patrick C. Fischer, Chairman of the Vanderbilt University Computer Science Department.

194. The three requirements for patentability, "which are taken from 35 U.S.C. §§ 101, 102, and 103," are novelty, utility, and nonobviousness. P. ROSENBERG, *supra* note 80, at 8-1.

195. Interview with Professor Patrick C. Fischer, Chairman of the Vanderbilt University Computer Science Department.

erly has defined software algorithms as procedures for solving mathematical problems and has characterized them as unpatentable natural laws. The Court's conceptualization of software algorithms is erroneous for several reasons. First, the Court improperly limited its definition of algorithms to procedures that solve mathematical problems even though people can use algorithms to solve virtually any problem. Second, the Court failed to recognize that not all software algorithms are equivalent to natural laws. Courts instead should view software algorithms, like other scientific processes, as lying on a continuum between clearly patentable and clearly unpatentable processes. Last, the Court equated software algorithms with unpatentable mathematical formulas without realizing that these algorithms satisfy its interpretation of the Patent Act's "process" definition. Thus, some software algorithms should qualify as patentable processes under section 101 of the Patent Act, and courts and legislators, therefore, should weigh the costs and benefits of extending patent protection to software programs.

Patent protection should extend to computer software only if the benefits of this protection outweigh the costs. Society would benefit in the following ways if patents protect software programs in the future: First, the stock of technological knowledge would increase in the software industry; second, this new knowledge would spread throughout the software industry and trigger new software innovation; third, new software innovation would help society use scarce resources more efficiently; and last, more and better applications for software products would arise and improve the quality of products that software helps to produce. These benefits collectively would improve the quality of life in America and contribute to an increase in America's gross national product.

Extending patent protection to computer software innovations, however, would force society to incur several costs. First, patent holders would receive the power to charge monopoly prices for the use and sale of patented software. Second, patent holders sometimes might misuse patent rights in an invention to the detriment of society. Last, the availability of software patents probably would cause software producers to flood the Patent Office and courts with patent claims; these claims would lead to exorbitant administrative and judicial costs.

We live in an age of widespread technological advance and proliferating computer prominence; therefore, the balance of these benefits and costs compels a conclusion favoring software patentability. The monopoly power and patent misuse costs of extending

patent protection to software algorithms are familiar problems that courts and patent administrators repeatedly have encountered and resolved throughout patent law history. In addition, the Patent Office probably could overcome the administrative burden of processing software patent applications by adding a small group of computer science experts to its administrative staff. Thus, extending patent protection to computer software would promote the technological growth of American society and improve the long term health of the economy.

JEFFREY S. GOODMAN*

* The author gratefully acknowledges the assistance of Harold G. Maier, Professor of Law, Vanderbilt University School of Law; George H. Sweeney, Associate Professor of Economics and Business Administration, Vanderhilt University; and Patrick C. Fischer, Professor and Chairman of the Computer Science Department, Vanderbilt University.