St. John's Law Review

Volume 74
Number 4 *Volume 74, Fall 2000, Number 4*

Article 2

March 2012

Better Living Through Software: Promoting Information Processing Advances Through Patent Incentives

Richard S. Gruner

BETTER LIVING THROUGH SOFTWARE: PROMOTING INFORMATION PROCESSING ADVANCES THROUGH PATENT INCENTIVES

RICHARD S. GRUNER*

Increasingly in our daily activities information is power, computers control information, and software programs[1] govern

* Registered patent attorney and a former inside counsel for the IBM Corporation. Professor of Law, Whittier Law School. Member of the New York and California state bars. LL.M., Columbia University School of Law; J.D., USC Law School; B.S., Caltech.

[1] Computer programs are comprised of instructions to computers dictating the sequence of information processing and storage steps the computers will take. These instructions are both technological and intellectual products. They are capable of achieving practical, technologically valuable results. Yet, software instructions are also intangible, intellectual constructs. One perceptive programmer described this dual character of software and some of its implications for software patents as follows:

> A computer program is [comprised of] the written instructions by a human being to tell a computer how to perform a particular task. As such, there are only two parameters—the input supply to the program and the expected output. Everything else is literally a figment of someone's imagination.
>
> This bears clarification. A computer program is the means of manipulating the internal data passed through a computer system. There is no requirement that the manipulations have any correspondence to the real world. In this, the real world, doing anything requires the expensive movement of people and goods from one point to another, the possible refinement of materials into other materials and the expenditure of energy and resources.
>
> Doing anything in a computer is merely the essentially cost-free movement of electron paths from one direction to another . . . , [and this creates] a world in which anything is possible.
>
> . . . .
>
> There are things that can be done within a computer program that cannot be done in the real world or would have undesirable consequences. As such, we should ask whether the patent rules which are designed to apply to real world conditions where doing something requires the expenditure of

computers.[2] Direct and significant links exist between software advances and lifestyle improvements.[3] Many of the benefits of

---

energy and resources should apply where the known rules of the universe do not apply. Because the entire design [of software] starts from scratch, the designer doesn't just get to play God, he is God.

United States Patent and Trademark Office, Public Hearing on Use of the Patent System to Protect Software-Related Inventions 4 (Feb. 10 & 11, 1994) (statement of Paul Robinson, Chief Programmer, Tansin A. Darcos & Company); *see also* Michael A. Dryja, *Looking to the Changing Nature of Software for Clues to its Protection*, 3 U. BALT. INTELL. PROP. L.J. 109, 111–12 (1995) (explaining the transformation in computer software programming between 1980 and 1995); David R. Syrowik, *Position Paper: Software Patents—Just Make A Good Thing Better*, 2 MICH. TELECOMM. & TECH. L. REV. 113 (1996) (visited Oct. 5, 2001) <http://www.mttlr.org/voltwo/syrowik.pdf> (stating that computer programs are included not only in software products, but also in most consumer and industrial products amenable to electronic control).

[2] Computer software has changed the fundamental economics of many business and social activities. As summarized by one analyst:

> The economic implications of the digital revolution extend far beyond the software industry. Any information-based industry will be subject to the same forces as it automates and moves on-line. Banking, for example, has traditionally been a diminishing-returns business. But as banking moves on-line, the work of processing customers' business is done by cheap computers, not expensive staff. The bank that has the greatest reach and can spread its fixed costs over the largest number of customers will be able to offer the best rates and deals, thereby attracting even more customers. Enter increasing returns.
>
> The same can be said for all sorts of service industries, from processing insurance-claims forms to managing inventories. Whenever computers and networks can greatly diminish variable costs, volume suddenly becomes all-important: the more the better .... "Everything's going software," says ... [Stanford economist Brian] Arthur, with a gleam in his eye.

Survey: The Software Industry, THE ECONOMIST, May 25, 1996, at 13–14.

[3] *See United States Patent and Trademark Office, supra* note 1 (statement of Stephen L. Noe, Intellectual Property Counsel, Caterpillar, Inc.).

> The practical impact of software and software-related products is felt in diverse personal and business activities. For example, software-related technology ... include[s] discreet software products like word processors or speaker timing computers, highly complex custom software that controls manufacturing systems and imbedded software that controls engines, anti-lock braking systems [and] perhaps your microwave oven.

Id.

In many settings, software-controlled computer systems have replaced earlier mechanical designs without necessarily being perceived as fundamentally different replacements by the users of the products involved. Where this substitution has improved the performance or price of products, consumers have generally benefited. The speed, effectiveness, and low cost of computer controls for physical devices and processes has extended the significance of computer programming and innovation into almost every engineering field.

The extension of computer applications into diverse fields makes the scope of patent protections for software technology particularly important. The profusion of

innovative software remain hidden and unappreciated.[4] Yet, with the pervasive use of computers, modern society is increasingly indebted to software innovators for the enjoyment of "better living through software."[5]

---

software-based innovations means that patent incentives for these innovations may produce benefits in an unusually wide range of fields. Additonally, parity between patent protections for software and older physical device designs may be desirable to balance software and physical device development incentives. For example,

> [w]hether an automobile engine is controlled by a camshaft or a microprocessor . . . makes little difference to the driver of that automobile who only cares that the engine run well and reliably. Patent policy should not be the factor that forces a manufacturer to choose which tool to use to control that engine. [S]oftware-related technology [should be treated] like any other technology within the scope of the patent system. Continued patent protection of software-related technology is important to the United State's industrial competitiveness.

Id.; *see also* Robert Greene Sterne & Edward J. Kessler, *An Overview of Software Copying Policies in Corporate America*, 1 J.L. & TECH. 157, 158–160 (1986) (studying the increased use of microcomputers in a corporate environment); Jeffrey S. Goodman, *The Policy Implications of Granting Patent Protection to Computer Software: An Economic Analysis*, 37 VAND. L. REV. 147, 175–76 (1984) (arguing that patent protection of software programs would encourage technological advancement, leading to an improved standard of living for the average American).

[4] This may stem, in part, from the difficulty of measuring the scope and effect of software usage. As noted by one industry analyst:

> Software is so abstract and ephemeral—just ones and zeros, recorded in tiny magnetic fluctuations or microscopic pockmarks—that its true economic significance is not always recognized. Computer hardware is much easier to quantify by the usual measures of the industrial age: chips, boxes, lorryloads. To use the language of Nicholas Negroponte, the MIT Media Lab's digital guru, computers are atoms. Software is bits. Atoms can be weighed, seen and felt. Bits, like genetic code, are inanimate—until they are placed into the right vessel. Then they use that vessel's machinery to create life.

Survey: The Software Industry, THE ECONOMIST, May 25, 1996, at 17; *see also* Jim Salter, *A Practical Approach to Claiming Software*, 14 SANTA CLARA COMPUTER & HIGH TECH. L.J. 435 (1998).

[5] This phrase is an updated version of the slogan "better living through chemistry," which the DuPont Corporation used for many years in its advertising. The substitution of software for chemistry in this updated version of the slogan is appropriate to reflect the nature of innovation and lifestyle improvements in our present society. Today, the focus of innovation has shifted from making new materials through better chemistry, to better managing existing materials and environments through improved information processing. The enhanced information processing, which has made this new type of innovation possible, is a product of improvements in computer hardware capabilities and further advances in software for applying computers to critical analyses and control tasks. *See* Samuel A. Guiberson, *The Challenge of Technology in the New Practice of Law*, 25 OHIO N.U. L. REV. 563, 582–83 (1999) (explaining the use of technology in courtrooms); *Survey: The New Economy*, THE ECONOMIST, Sept. 23, 2000, at 6–7 (discussing examples of computer technology's impact on daily living).

As computer capabilities improve and computer-based devices and processes proliferate,[6] our reliance on software and improvements in software engineering is likely to increase as well.[7] Information processing by computers is the defining technology of our age and software design is its guiding force. Society's stake in computer processing improvements implies a corresponding stake in legal standards and constructs that encourage software advances.

In older fields concerned with the design of physical items or processes, the primary legal means for encouraging engineering progress have been patent incentives.[8] Utility patents provide inventors of new and useful devices and processes with temporary control over the use, making and sale of their inventions.[9] Patent laws provide this control to potential inventors as an incentive to invest resources in discovering new advances.[10]

---

[6] "With revenues of than more $200 billion and a growth rate of some 13% a year, software is one of the world's largest and fastest-growing industries." *Survey: The Software Industry*, THE ECONOMIST, May 25, 1996, at 4. These figures focus on companies that produce software products. Expenditures on software development are also large within companies that use software to achieve operational improvements rather than as the basis of marketable products. For example, AT&T, not a major seller of software itself, spent approximately $1.8 billion on software development in 1994. This represented 60% of the company's total research and development budget. *See United States Patent and Trademark Office, Public Hearing on Use of the Patent System to Protect Software-Related Inventions* 7 (Jan. 26 & 27, 1994) (statement of William Ryan, AT&T attorney) <http://www.uspto.gov/web/office/com/hearings/software/sanjose/sjhrng.pdf>.

[7] *See id.* at 24 (statement of Jerry Baker, Senior Vice-President of the Oracle Corp.).

[8] *See* Jeffrey S. Goodman, *The Policy Implications of Granting Patent Protection to Computer Software: An Economics Analysis*, 37 VAND. L. REV. 147, 147 (1984) (analyzing the benefits and costs of granting patent protection to computer software).

[9] *See* 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, 71–82 (1996) (analyzing the economic benefits for investors with the existence of intellectual property protection).

[10] Patents are not available for every technological advance, but only for new technological discoveries which are a significant step or non-obvious "leap" beyond prior technological knowledge. *See* 35 U.S.C. § 103 (2001). An invention is deemed to reflect such a leap under current patent law standards only if persons working on similar engineering projects, e.g., engineers with average knowledge and skills in the field of the invention who are fully informed about publicly available technology in that field, would probably have been unable to develop the same invention with a minimal amount of analysis and experimentation. *See* Graham v. John Deere Co., 383 U.S. 1 (1966); Goodman, *supra* note 8, at 175–76.

In addition to promoting the discovery of new inventions, patent controls over newly discovered but unperfected and uncommercialized technologies can encourage inventors and companies to invest substantial sums in designing products based on new and untested technologies and in marketing those products. Patent incentives favoring the creation and marketing of new innovations are aimed at expanding consumer choices. Patent incentives help to promote greater numbers and diversity of technological discoveries and more complete disclosure and commercialization of those discoveries.[11]

Recently, substantial controversy has raged over whether the patent system should govern software design.[12] Three types of arguments have been asserted against software patents. First, all patents governing intellectual constructs like software are undesirable because they restrict free access to fundamentally important modes of analysis and information processing tools.[13] Second, software patents are particularly undesirable because software development differs from earlier forms of engineering in ways that cause patent restrictions to be burdensome limitations rather than helpful incentives.[14] Third,

---

[11] *See* Goodman, *supra* note 8, at 175–76.

[12] Patents on information processing innovations are hardly new. Rather, patents have played an important role in encouraging information processing advances for many years. Some early patents on information processing innovations significantly predate the development of the computer. For example, Samuel Morse's patent on his design for the telegraph claimed, among other features, the use of a system of dots and dashes as a means of transmitting and recording information about letters and words. In essence, this was a patent on an information processing method and storage system. Insofar as it covered a specialized method for manipulating a physical device to achieve an information processing result, the Morse patent has been described as an early software patent. *See United States Patent and Trademark Office, supra* note 6, at 90 (statement of Roger Schlafly); *see also* Steven L. Friedman, *Don't Mess with Good Patent Law*, THE NAT'L L.J., Mar. 27, 2000, at A26. (arguing that the invention of novel e-commerce business methods or software applications should not be denied the protections of the patent laws); Charles R. McManis, *Patent Law and Policy Symposium: Re-Engineering Patent law: The Challenge of New Technologies*, 2 WASH. U. J.L. & POL'Y, 1 (2000).

[13] *See, e.g.*, Mark A. Haynes, *Commentary: Black Holes of Innovation in the Software Acts*, 14 BERKELEY TECH. L.J. 567, 568 (1999) (discussing the need for flexibility with patents so that patents do not block innovation within software arts).

[14] *See, e.g.*, John M. Griem, Jr., Note, *Against a Sui Generis System of Intellectual Property for Computer Software*, 22 HOFSTRA L. REV. 145, 155 (1993) (outlining the argument that "software should be treated differently than earlier technologies because the nature of computer software allows its creators to write,

whatever the merits of software patents in general, many recently issued software patents illegitimately limit public access to the software involved because the United States Patent and Trademark Office (PTO) lacks the resources to properly assess software patents.[15] One commentator has summarized these arguments in the following succinct terms: "patents are bad, software patents are bad, and bad software patents are bad."[16]

This article argues that these concerns are misplaced. It describes the substantial policies favoring patent rewards and controls for innovative software. To identify the potential benefits of software patents, the analysis here first examines the benefits of patent protections in other technological areas. The availability of these same benefits from software patents is then assessed. Based on the conclusion that traditional patent benefits can be realized through software patents, this article advocates substantial patent protections for new and innovative software designs.

I. POTENTIAL IMPACTS OF PATENTS ON SOFTWARE DEVELOPMENT

While there is no universally accepted definition of technological design processes, these processes generally involve the creation of man-made materials, devices, or procedures that are useful in solving practical problems faced by individuals and businesses.[17] Patents encourage individuals to develop, disclose,

---

market and sell software in less time but . . . if all these technologies were protected by patents, programmers would be unable to work freely"); Randall M. Whitmeyer, *A Plea for Due Process: Defining the Proper Scope of Patent Protection for Computer Software*, 85 NW. U. L. REV. 1103, 1127 (1991) (explaining that "patents would unduly restrict the incremental, building-block approach common in the software industry, and would probably accelerate a trend toward oligopoly in the software industry").

[15] *See* Sandra Szczerbicki, *The Shakedown on State Street*, 79 OR. L. REV. 253, 274 (2000) (explaining in regard to business method patents that the PTO's software examiners, "many of whom are young engineers, simply do not have sufficient expertise to evaluate patents which cover internet business models" and that "[s]ince software was not considered patentable until recently, there is no database covering the first thirty years or so of software development to help agencies make novelty and obviousness judgments"). Szczerbicki concludes that "[a]s a result of these inadequate searches, the PTO is issuing, and will continue to issue, invalid patents." *Id.*

[16] *United States Patent and Trademark Office, supra* note 6, at 56 (statement of Ronald S. Laurie of the law firm of Weil, Gotshal & Manges).

[17] Technology involves "the new things you can do when you have found

and popularize new technological inventions that represent substantial departures from prior knowledge.[18] An assessment of the desirability of applying patent incentives to software development must start with an understanding of the design steps and decisions undertaken in software development. From this understanding, we can assess whether there are features of software design activities that can be influenced through patent incentives.

For purposes of this analyses, computer software is comprised of instructions given to a computer to cause the computer to undertake a sequence of information processing steps.[19] Typically, computer processing is undertaken to

---

something out." RICHARD P. FEINMAN, THE MEANING OF IT ALL 4–5 (1998). Expansion of technological knowledge occurs through engineering, which is "the practice of organizing the design and construction of any article which transforms the physical world around us to meet some recognized need." G.F.C. ROGERS, THE NATURE OF ENGINEERING: A PHILOSOPHY OF TECHNOLOGY 51 (1983); *see also* COMPUTER DICTIONARY 384 (2nd ed. 1994) (technology is the product of the "application of science and engineering to the development of machines and procedures in order to enhance or improve human conditions, or at least to improve human efficiency in some respect").

In some cases, new technological designs follow directly from scientific discoveries as newly understood scientific ideas or principles are used to specify the features of new types of useful inventions. New scientific understanding, however, is not needed to produce new inventions. In some instances, new technological designs involve the application of older scientific or engineering principles in new ways to solve long-standing practical problems or to solve new practical problems stemming from new social practices. *See* WALTER G. VINCENT, WHAT ENGINEERS KNOW AND HOW THEY KNOW IT 12–13 (1990).

In addition to having varying sources, new technology can have a variety of functional impacts on users: "Technology makes it possible to do something never done before (the airplane), to do mechanically something previously done manually (the sewing machine), or to do more effectively something previously done mechanically (the repeating rifle)." DAVID FREEMAN HAWKE, NUTS AND BOLTS OF THE PAST: A HISTORY OF AMERICAN TECHNOLOGY, 1776–1860 8 (1988).

[18] *See, e.g.*, Graham v. John Deere Co., 383 U.S. 1, 5–19 (1966) (providing a history of the patent system to meet the constitutional command of promoting "useful" arts); Florida Prepaid Postsecondary Educ. Expense Bd. v. College Sav. Bank, 527 U.S. 627, 650 (1999) (Stevens, J., dissenting) (explaining that patent statutes should be interpreted so that they comport with "constitutional goals of stimulating invention and rewarding the disclosure of novel and useful advances in technology").

[19] One observer has given a more lyrical definition of computer software:

> Computers are to computing as instruments are to music. Software is the score, whose interpretation amplifies our reach and lifts our spirit. Leonardo da Vince called music "the shaping of the invisible," and his phrase is even more apt as a description of software. As in the case of music, the invisibility of software is no more mysterious than where your

transform one type of information, e.g., information about items purchased with a credit card, into another type of information, e.g., billing records specifying the amount due on a credit card account. Computer software mediates between the functional needs of a computer user and the characteristics of a particular information processing problem.[20] Instructions embedded in the software tell the computer how to apply its capabilities to solve the problem. Software quality often turns on how well computer capabilities are applied to address an information processing problem thoroughly and efficiently.

A. Targets of Patent Incentives—Software Design Choices

Software advances are based on new information processing ideas and insights that are embedded in corresponding computer instructions. At least three different types of design choices and corresponding opportunities for innovation are present in the design and implementation of a software-based device or process. These design choices involve the identification of information concepts or relationships that are relevant to the practical problem the device or process will solve (hereinafter "conceptualization"), the specification of information processing steps or data structures that will capitalize on these concepts or relationships (hereinafter "coding"), and the connection of these information processing steps to a broader physical context to achieve a useful result (hereinafter "external linkage"). Patent incentives may influence creative efforts and design activities concerning each of these types of design choices.

1. Conceptualization

Innovation in the conceptualization stage of software design involves seeing new ways that information can be organized and processed by a computer. New designs can either improve existing information processing methods or implement entirely new modes of information processing.[21] This type of innovation

---

> lap goes when you stand up. The true mystery . . . is how so much can be accomplished with the simplest of [software programming] materials given the right architecture.

Alan Kay, *Computer Software*, SCI. AM., Sept. 1984, at 53.

[20] *See id.* at 54 (describing the information processing instructions of computer software as means to mediate between the needs and capabilities of a computer user and the practical tasks to which a computer is applied).

[21] *See United States Patent and Trademark Office*, *supra* note 1, at 31

involves diagnosing problems and constructing solutions in light of a computer's information processing capabilities.[22] A developer focusing on this type of software innovation might seek to create an improved definition of the information processing features of a practical problem. Alternatively, a developer may seek to create a method of bypassing past information processing deficiencies. A developer might also seek a new information processing technique for solving a practical problem.

The conceptualization phase of a software design project may involve a combination of insights into the characteristics of a practical problem, the capabilities of computer-based information processing and storage to solve all or part of the problem, and the proper means to match a particular information processing approach to the characteristics of the problem. By analyzing these aspects of the problem, a software developer can identify the necessary information processing steps for addressing and solving a practical problem, and thereby produce a new software design concept with practical utility. There is no question that many new and important types of software involve information-processing advances at these basic conceptual levels.[23]

(statement of Leonard Charles Suchyta, Assistant Vice President, Bell Communications Research, Inc. (Bellcore)) (noting that the software-related inventions that Bellcore seeks to patent typically involve innovations in the conceptualization and definition of information processing functions and relationships, rather than new types of computer programming code).

[22] Impatience with the functional limitations of current practices and attempts to understand and solve problems with those practices are common precursors to inventive efforts. As noted by leading technology analyst Henry Petroski:

> Regardless of their background and motivation, all inventors appear to share the quality of being driven by the real or perceived failure of existing things or processes to work as well as they might. Fault-finding with the made world around them and disappointment with the inefficiency with which things are done appear to be common traits among inventors and engineers generally.

HENRY PETROSKI, THE EVOLUTION OF USEFUL THINGS 38 (1992).

[23] *See, e.g.*, Kay, *supra* note 19, at 57 (describing how developers of the first computer spreadsheet program recognized that past problems in correcting financial reports could be overcome by a new conceptual approach to changing financial reports in which a change in one financial figure is propagated by computer information processing into updates of other related figures).

2. Software Coding

Even where a method of information processing is well understood and simply reused in new software, software innovations may result from new ideas about how to translate the old method into computer processing steps and corresponding programming instructions or "software code."[24] Innovations of this sort are sometimes economically significant because efficiency gains in repeated computer processing can be very valuable. This value stems from the frequent reuse of computer operations to accomplish useful tasks such that even a modest gain in the efficiency or effectiveness of each iteration of processing can result in a large improvement in the overall operation of a program.

Gains in processing speed are not the only possible innovations in computer programming. The speed of producing and checking or "debugging" the code itself is an important consideration because labor costs for software coding are often

---

[24] By programming a computer to undertake a specific pattern of information processing, a software developer is assembling the various information processing capabilities of the computer into a sequence which accomplishes a desired task. Even where the beginning and end points of this sequence are well known—because the overall information processing task being performed is an old one—the particular information processing sequence chosen by the computer programmer to accomplish the task may be new.

Innovation in the selection of computer processing paths—through the selection of statements in a particular computer programming language defining the paths—may have a large impact on system performance. Greater efficiency and effectiveness in information processing can result if software coding choices are made so as to invoke only so many of the computer processing steps as are necessary to produce some information processing result. The goal of this coding effort is to select and sequence portions of the capabilities of a general purpose computer to create the equivalent—in the resulting computer hardware-software combination—of a new, specialized information processing machine with capabilities and actions that are tailored to the information processing task at hand. This capability of software designers to define new machines by careful programming has been described by one observer as follows:

> A [programming] language and the software that "understands" it can totally remake the computer, transforming it into a machine with an entirely different character. The hardware components of a typical computer are registers, memory cells, adders and so on, and when a programmer writes in the computer's native language those are the facilities he must keep in mind. A new language brings with it a new model of the machine. Although the hardware is unchanged, the programmer can think in terms of variables rather than memory cells, of files of data rather than input and output channels and of algebraic formulas rather than registers and adders.

Lawrence G. Tesler, *Programming Languages*, SCI. AM., Sept., 1984, at 70.

high. A new means of reusing existing codes or otherwise reducing the scope or complexity of coding efforts can be a valuable advance in the context of computer software development. These types of innovations concerning programming techniques and tools are also important targets of software design innovations.

Finally, programming method changes that improve the reliability of coding processes are another important type of software design innovation. Large software projects frequently suffer from reliability problems because the computer code involved is so complex that it is hard to test all of the operations and combinations of operations that a computer will perform while running under the control of the code. Hence, programming design advances may have value because they assist programmers in ensuring that computer operations are complete, compatible with each other, and effective in their intended functional roles.

3. External Links to Inputs and Outputs

Even if a series of information processing steps has been implemented in computer code, these steps must still be linked to an external context in order to produce a practical result.[25] This external linkage typically involves specifying means to input information for computer processing and further specifying means to output the results of that processing. External linkage of computer processing may also involve using the outputs of computer processing to archive further physical results such as the control of a machine or manufacturing process.

The development of data input means for a computer system

---

[25] This process entails the specification of design elements that apply the information processing power of a computer programmed in a particular way to achieve a practical end. Typically, this will be accomplished through testing tentative designs of computer-based devices to determine their practical effectiveness and efficiency. In the engineering of computer-based devices, as in other engineering contexts:

> [A design process] typically involves tentative layout (or layouts) of the arrangement and dimensions of the artifice, checking of the candidate device by mathematical analysis or experimental test to see if it does the required job, and modification when (as commonly happens at first) it does not. Such procedure usually requires several iterations . . . Numerous difficult trade-offs may be required, calling for decisions on the basis of incomplete or uncertain knowledge.

VINCENT, *supra* note 17, at 7.

may be as simple as deciding to use standard data entry devices such as a keyboard or computer mouse. In some computer systems, elaborate devices or processes for gathering information about particular physical phenomena or environments will be necessary. Significant design efforts may be devoted to the definition of these data gathering components of computer systems. These design efforts typically are aimed at creating measurement devices or processes that can provide data about a physical setting in quantities and forms that can be analyzed by a computer to produce new information concerning the physical setting.

For example, a computer-based light meter might include a data gathering component in the form of a silicon cell that measures light intensity, a further component that translates the intensity measurements into digital data suitable for computer analysis, then a specially programmed computer that analyzes this data to evaluate the best camera exposure for a scene. Beyond the development of the programming incorporated in this device, considerable time may be spent on designing the light measurement and analog to digital data conversion components of this device and in making these three components work together in the desired application.

Similarly, the development of means for outputting computer-processing results may also involve substantial output design problems and corresponding difficulties in producing practical designs. Usually, computer output linkages to an external environment will involve one of two types of design elements. First, information processing results can be used to manipulate an external activity or device. For example, a computer might be used to control a rubber molding process. This might be accomplished by having the computer analyze mold temperature data to estimate the proper time for completion of a molding procedure and to send signals from the computer to terminate the molding process at the proper time.[26] Second, output designs may be developed to display computer-processing results in ways that are easy for humans to

[26] This type of software output was present in the invention found to be patentable subject matter by the Supreme Court in *Diamond v. Diehr*, 450 U.S. 175 (1981). In *Diehr*, the Supreme Court held that a physical and chemical process for molding precision synthetic rubber products was patentable subject matter, rather than an attempt to patent a mathematical formula. *See id.* at 192.

understand.[27] These types of "user interface" features of software designs have an important impact on the functionality and usefulness of software. User interface designs are therefore important targets of software development efforts.[28]

New input and output components are present in many computer-based inventions. A particular invention may include both these types of design elements. For example, a new knife design might use an array of light sensors to detect the edge of an item being cut, a computer to analyze the sensor data to identify the probable location of the edge and to predict the path of cutting that will remove a narrow additional slice from that edge, and further electronic linkages to allow the output of the computer processing to control the physical movement of a knife blade along the selected cutting path. The development of this invention would entail specification of both the edge-measuring input components and the knife-controlling output linkages needed to apply the new computer information processing involved in the invention to an external cutting context.

B. The Patentability of Software as a Separate Invention

Many software innovations have been marketed to consumers as part of broader inventions in which the new software was merely a component of a broader device or process.[29] In these broader inventions, the new software typically controlled a physical device feature or process step to

---

[27] Functional and visually attractive user interfaces—often referred to as Graphical User Interfaces (GUIs)—are frequently difficult to design and implement in an effective manner. Software developers who specialize in the design of these useful components of software products argue that patent restrictions on innovative GUIs are justified to spur and protect the significant development efforts required in this area. *See United States Patent and Trademark Office, supra* note 1, at 42–44 (statement of Timothy Scanlon, Human Interface Specialist, Allen-Bradley Company).

[28] *See* Kay, *supra* note 19, at 54 (recognizing the emphasis placed on interface designs because they are the most recognizable aspects of the computer from a user's perspective).

[29] *See, e.g., Diamond*, 450 U.S. at 175 (discussing software included in a rubber molding process); Parker v. Flook, 437 U.S. 584, 586 (1978) (discussing software included in a fire alarm triggering system); State St. Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 1371 (Fed. Cir. 1998) (examining computerized accounting system used to manage mutual fund investment structures); Arrhythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053 (Fed. Cir. 1992) (describing method of analyzing electrocardiographic signals in order to determine certain characteristics of heart functions).

produce a useful result. In some instances, these broader inventions were updated versions of earlier purely physical designs in which software and computer controls were added to improve the earlier versions of the same devices or processes.[30] In other settings, the new inventions were complete redesigns and substitutes for earlier devices or processes that produced similar results. In this type of invention, the new software-based version of the device or process might have few features in common with its earlier mechanical counterpart, yet the overall new invention would still be analogized to and analyzed in terms of its earlier counterpart.[31]

Many legal controversies involving software innovations have focused on these sorts of hardware/software combinations. Frequently, courts assessed the patentability of these inventions based on the overall features of the inventions rather than the features of the software alone. Where inventions included physical components and results that were similar to those of non-computer based designs and processes, courts had little difficulty in finding the software-controlled devices or processes to be updated versions of their earlier, purely mechanical counterparts.[32] Since the purely mechanical versions of the devices or processes had sufficient physical components and results to be patentable subject matter, and since the new software-controlled designs possessed similar components and results, many physical devices and processes involving software controls were found to be patentable subject matter.[33]

---

[30] *See, e.g.*, *Diehr*, 450 U.S. at 179 n.5 (involving the addition of computer analyses to an existing rubber molding process to better assess the progress of the molding and to assess the optimal time to open a mold).

[31] For example, an invention might entail the creation of a digital speedometer that operates internally through a variety of computer processing steps, which are nothing like the physical operations of earlier mechanical speedometer designs, but perform the same overall function as those earlier designs. In the eyes of car designers and consumers, the digital and mechanical versions of the speedometer may be considered substitutes without much thought as to how their respective internal activities are carried out. These two types of devices may also be analogized and treated similarly in patent analyses even though the creation of one—the digital speedometer—turns primarily on the designing of intangible information processing relationships and the other turns on the designing of physical components and operational relationships between those physical components.

[32] *See, e.g.*, *In re* Alappat, 33 F.3d 1526, 1543–54 (Fed. Cir. 1994) (stating that patent claim subject matter must be viewed in terms of the totality of its components).

[33] *See, e.g.*, Schlafly v. Public Key Ptrs., No. 94–20512 SW, 1997 U.S. Dist.

The Supreme Court used this type of reasoning to assess the patentability of a software-controlled invention in *Diamond v. Diehr*.[34] In *Diehr*, the Court considered the patentability of a rubber molding system in which computer controls were added to a rubber molding process.[35] The Court concluded that, even though the only innovation in this new design was the addition of computer information processing to interpret mold temperature data and trigger the ending of a molding activity by opening a mold, the resulting process was patentable subject matter because the invention taken as a whole possessed the sorts of physical structures, e.g., the structures of the rubber mold, and results, e.g., the features of the molded item, required of a patentable invention.[36]

This sort of patentability analysis—emphasizing the features of a claimed invention beyond the software-implemented controls included in the invention—provides little guidance concerning the patentability of true software inventions. For example, this approach is not useful in assessing advances in which software implements information processing that is valuable without being translated, at least immediately, into a useful physical activity or result. Such pure software advances can usually not be assessed in terms of analogies to earlier mechanical counterparts. Judicial standards or analytic approaches that depend on these analogies or on examination of other physical features of inventions have no applicability in determining the patentability of pure software inventions.

Newer standards governing the patentability of pure software inventions are needed because such inventions are increasingly common and useful. Many software innovations entail advances that are valuable as information processing tools. The valuable features of these software innovations may have little to do with the physical way they are implemented or the physical contexts in which they are used. Rather, modern software innovations can create information processing systems

---

LEXIS 15251, at *12–15 (N.D. Ca. Aug. 22, 1997) (concluding that claims of a patent related to cryptography made use of known physical structures; thus, holding that the patent did not merely monopolize a mathematical concept, but rather applied a mathematical calculation to existing hardware to produce a useful and tangible results).

[34] 450 U.S. 175 (1981).

[35] *Id.* at 177.

[36] *Id.* at 188–89.

that are, in essence, computer-implemented information analysis tools that can be applied in diverse physical contexts and roles.[37]

Diehr provides little guidance regarding the patentability of these software-based information analysis tools because the Court's analysis did not describe the minimum features of patentable software. While the Court confirmed in *Diehr* that the overall invention at issue in that case possessed sufficient tangible features and results to qualify as patentable subject matter, it did not address the more interesting question of whether some of the software sub-components of the rubber molding process at issue might have been separately patentable.[38] Indeed, a question remains as to whether the software controlling this process or a computer programmed to undertake the sequence of computer processing involved in the process are separately patentable.

The advantage to an inventor in obtaining patents on these separate invention components is that the patents would control the use of the same software or computer-processing sequences in other application contexts. If available, these types of patents on basic computer-processing components will often be particularly significant sources of control over later computer applications.

C. *The Need for Software Patents Over and Above Other Intellectual Property Protections for Innovative Software*

Computer software is an unusual type of intellectual property in that software has both descriptive and functional

---

[37] *See, e.g.*, Alan L. Durham, *"Useful Arts" in the Information Age*, 1999 B.Y.U. L. REV. 1419, 1513–14 (differentiating between hardware, which has physical manifestations and algorithms, which are not visually discernible yet perform important tasks in electronic media like virus scans and web searches).

[38] *See generally* Maximilian R. Peterson, Note, *Now You See It, Now You Don't: Was it a Patentable Machine or an Unpatentable "Algorithm"? On Principle and Expediency in Current Patent Law Doctrines Relating to Computer Implemented Inventions*, 64 GEO. WASH. L. REV., 90, 132 (1995) (stating that there are no "clear guiding principles of public policy" to determine the patentability of computer implemented devices, and recognizing that prevailing law denies effective protection for such developments); Robert C. Laurenson, *Computer Software "Article of Manufacture" Patents*, 77 J. PAT. & TRADEMARK OFF. SOC'Y 811, 812 (1995) (characterizing the *Diehr* decision as limited to a narrow class of software); Aryeh S. Friedman, *Law and the Innovative Process: Preliminary Reflections*, 1986 COLUM. BUS. L. REV. 1, 9 (stating that the majority's decision was very narrow, and "simply held that an otherwise patentable industrial process which employed a computer system to perform essential calculations was not thereby rendered unpatentable").

features. As a descriptive/functional hybrid, computer software has proven to be a difficult target of intellectual property protections.[39] Software that is capable of directing a computer to complete a useful task may qualify for several varieties of intellectual property protections.

As a means of describing a set of computer operations, computer software is a specialized form of human expression that qualifies, like other types of human expression, for extensive protections under copyright laws.[40] These laws generally protect against the copying of original software statements, such as the copying of programming code.[41]

---

[39] For an overview of the full range of issues surrounding intellectual property protections for computer software, see RAYMOND J. NIMMER, THE LAW OF COMPUTER TECHNOLOGY ¶ 1.01–5.13 (2d ed. 1996); *see also* Jane Rolling, *No Protection, No Progress for Graphical User Interfaces*, 2 MARQ. INTELL. PROP. L. REV. 157, 174–81 (1998) (noting the unique nature of software and its legal impact); BERNARD A. GALLER, SOFTWARE AND INTELLECTUAL PROPERTY PROTECTION: COPYRIGHT AND PATENT ISSUES FOR COMPUTER AND LEGAL PROFESSIONALS 67 (1995); Melissa Hamilton, Note, *Software Tying Arrangements Under the Antitrust Laws: A More Flexible Approach*, 71 DENV. U. L. REV. 607, 612 (1994) (stating that the degree of copyright protection for software developers "is still open for interpretation").

[40] Software can qualify for copyright protections because a programming code is a type of human expression that serves several communicative functions. For example, computer software can serve as a vehicle for human-to-human communication. This occurs where software is used as a means for one programmer to record a sequence of computer processing steps for understanding by later programmers. This is a specialized, but traditional form of human-to-human communication.

Even if we ignore this human-to-human usage of computer software code and focus exclusively on its more important function of dictating computer processing steps, a type of human-originated communication is present. In this role, software serves to communicate information from a programmer to one or more computers. Under United States copyright law, this type of machine-oriented communication is treated as protectable expression on par with more common human-to-human writings. The role of a human as the author of such communications is sufficient to qualify programming code for copyright protection. This protection is granted in order to reward and encourage the efforts of programmers, notwithstanding the lack of numerous human readers for the programmers' software. *See, e.g.*, Cable/Home Communication Corp. v. Network Productions, Inc., 902 F.2d 829, 843 (11th Cir. 1990); Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521, 525 (9th Cir. 1984); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1254 (3d Cir. 1983); *see generally* PAUL GOLDSTEIN, COPYRIGHT § 2.15.2 (1989).

[41] Some cases have also extended copyright protections to prevent the copying of a program's functional "look and feel." *See* DONALD S. CHISUM & MICHAEL A. JACOBS, UNDERSTANDING INTELLECTUAL PROPERTY LAW § 4C[2][d] (1992). Recent cases, however, have severely limited this type of copyright protection for the functional attributes of programs. These cases have restricted copyright protections to the prevention of copying of aspects of program expression such as programming

Copyright protections, however, will generally not bar a party from replicating the functional characteristics of another party's software code through reprogramming. This makes copyright laws of limited value in protecting a software developer against the reuse of new methods of information processing embedded in computer software.[42]

Computer software also has functional characteristics that may be protectable under patent laws because it is capable of directing computers to undertake useful tasks. This protection may be extended on the basis that software is, in essence, a functionally significant component of a specially designed information processing machine. The overall machine can be realized by combining the software with a general purpose computer to undertake a useful function. [43] Just as an improved design for a physical cam which controls the motion of a machine in a beneficial way can be patented separately as a useful component of the machine, so too may new software with functional implications in controlling computer processing be patentable separately from the computers that the software controls.[44]

---

commands or details of internal program organization and structure. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–16 (1st Cir. 1995) (holding that Lotus 1-2-3 menu command hierarchy was uncopyrightable subject matter because it was a method of operation and foreclosed from copyright protection by 17 U.S.C. § 102(b)).

[42] "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b) (1994). *See also* Autoskill Inc. v. National Educ. Support Sys., Inc., 994 F.2d 1476, 1491 (10th Cir. 1993) (stating that it is a basic premise of copyright law that a copyright only protects the expression of an idea not the idea itself); Rubin v. Boston Magazine Co., 645 F.2d 80, 82 (1st Cir. 1981) (noting that merely an idea does not garner copyright protection).

[43] *See, e.g.*, Tesler, *supra* note 24 (describing how a computer program can "remake [a] computer, transforming it into a machine with an entirely different character"); Anthony J. Mahajan, *Intellectual Property, Contracts, and Reverse Engineering After PROCD: A Proposed Compromise for Computer Software*, 67 FORDHAM L. REV. 3297, 3297 (1999) (indicating that when software is run on a computer the computer becomes a different apparatus because it is lead by the directions of the program); Rinaldo Del Gallo, III, *Are "Methods of Doing Business" Finally Out of Business as a Statutory Rejection?*, 38 J.L. & TECH., 403, 434–435 (1997) (stating that "[s]oftware simply transforms the general purpose computer into a specific purpose computer").

[44] Software may be the most important design component in transforming a general purpose computer into a useful device in a particular application context.

The extension of patent protections to computer software standing alone has several important implications for software developers. First, unlike copyright protections that may attach to the same software, a patent covering a particular software product will prevent another party from independently developing and using functionally similar software during the life of the patent.[45] Second, another software developer will not be able to reuse the first developer's patented innovation by simply analyzing the first developer's software and replicating the innovative information processing sequences in new software code.[46] If the second developer wishes to replicate patented functional features of the software, the second developer will need to gain a license from the patent holder with appropriate compensation for the use of the first party's invention. Finally, patent protections for innovative software may promote

---

One software industry analyst has described the importance of software's computer-transforming role as follows:

> The beauty of the computer—indeed, the basis of its world-changing success—is that it is totally unspeciali[z]ed. A toaster has just one job: it toasts. A PC, on the other hand, is more like a blank sheet of paper: it is a generic device that serves as a carrier and a repository for ideas of any sort, from "Rebel Assault II" to "Richard III." Partly for this reason, the hardware and software worlds have grown up very differently. Hardware exists to do the bidding of software, not the other way around. Both can be good businesses to be in, but on balance software has the economic edge.

The Software Industry Survey, THE ECONOMIST, May 25, 1996, at 17.

[45] A second developer will not be free to independently develop and adopt a program that is similar to patented software since independent development is not a defense to patent infringement. *See* 35 U.S.C. § 154 (1994); *see also* NIMMER, *supra* note 39, ¶ 2.01 (noting that in comparison with copyright law, patent protections for software establish "a relatively stronger property right regime that gives exclusive rights against entirely independent development").

[46] This reuse of the functional details of the first developer's work would not be restricted by copyright protections on the first developer's work since it does not involve the copying of the first developer's protected expression. However, patent protections concerning the functional features of the first work would prevent unauthorized reuse of those functional features in additional software created during the life of the patent. *See* Note, *Computer Intellectual Property and Conceptual Severance*, 103 HARV. L. REV. 1046, 1057–58 (1990) (noting the lack of copyright protections for the functional features of UNIX program code); Pamela Samuelson, *Brief Amicus Curiae of Copyright Law Professors in Lotus Development Corp. v. Borland International, Inc.*, 3 J. INTELL. PROP. L. 103, 132 (1995) (indicating that Congress intended to exclude the functional results of a program from copyright protection); 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, 237 (1996) (noting that the goal of copyright law is to protect communicative subject matter while the purpose of patent law is to protect functional subject matter).

applications of software that are broader than if the software was freely usable. Since a developer of a new, patented information processing method imbedded in software will have control over all applications of the same software for the life of the patent, the patent holder has an economic interest in seeing that the patented software is used in all of its possible applications, even in fields beyond the setting where the software was first developed. Since the patent holder can count on collecting a portion of the resulting profits or licensing fees, the patentee will be encouraged to spread the use of the patented software to new fields by either developing applications in those fields or encouraging and licensing others to do so.

In sum, the dual expressive and functional characteristics of computer software should cause innovative software to qualify for both copyright and patent protections. Patent protections should apply because software dictates computer functions. Copyright protections should apply because software contains detailed expression by human authors that describes and communicates computer instructions.

The availability of both copyright and patent protections for a particular software program is desirable because it encourages two levels of software innovation. As new information processing ideas and applications are studied or discovered, the promise of patent rights will encourage research into related computer implementations and will promote public disclosures and popularization of functionally significant computer processing methods. In these ways, patent protections for new software will promote the sort of increased public access to new and useful tools—in the form of software-based products and services—which the patent laws were designed to achieve.

At the same time, the opportunity to gain copyright protections and rewards may create incentives for programmers to produce more efficient or effective software implementations of a particular method of information processing.[47] Where a method is unpatented, software developers can compete freely to produce the best copyrighted code for achieving a software implementation of that method. Similar competition may prevail even if an innovative computer-processing method is

[47] *See United States Patent and Trademark Office, supra* note 1, at (statement of Joseph Hofstader, League for Programming Freedom).

patented. When the holder of a patent on a computer-processing method is willing to issue several nonexclusive licenses to software developers allowing the developers to implement the patented method in their software, the resulting software products will each be separately copyrightable because they will each utilize original code to implement the patented method. Where one of these copyrighted products is significantly more effective or efficient than others in carrying out or applying the patented method, the author of this superior version of the software will be able to use his or her rights under copyright laws to control the marketing of the superior software product and gain associated rewards for the value of the software involved. Permitting copyright protections for programming code that implements patented software designs may therefore encourage a dimension of competition in programming implementations that is separate from competition to develop more fundamental information processing designs.

A similar type of two-level competition already applies to other types of inventions. When a practical problem arises in a given field, various engineers may seek to develop product or process designs to solve the problem. If one of them reaches a solution and gains a patent for it, this patented invention may be the subject of efforts by others to describe the patented invention in texts or other expressive works. Descriptive works of this sort are copyrightable. To the extent that several authors attempt to describe the new invention, their various writings compete with each other in a way that is independent of the prior competition that led to the development of the invention. While the efforts of both the engineers who compete to design the original invention and the authors who compete to describe it revolve around the invention itself, no particular legal problem is raised by letting these two types of related competition go forward. Indeed, the separate incentives of the patent and copyright laws encourage two distinct and independently desirable dimensions of competition.

In comparison with these sorts of separate creative efforts by invention developers and invention describers, a programmer who seeks to incorporate a patented software invention in a new set of software is attempting to both describe and implement the invention. The programmer is both an invention user and an invention describer. As a user of the patented invention, the

party writing the new software must obtain a license to do so from the patent holder before that party can go on to compete in describing that invention in the new software as effectively and efficiently as possible.[48]

This sort of linkage between describing and implementing an invention may be peculiar to functional/descriptive hybrids such as computer software. The rarity of this linkage, however, does not undercut the desirability of interpreting patent and copyright laws so as to encourage separate competition over the development of innovative computer software designs and the creation of specific software expressions for implementing those designs.

II. SHOULD SOFTWARE BE PATENTABLE?

Whether or not software innovations should be patentable depends on the net public benefits to be realized from software patents. Where granting software developers patent controls over particular new software designs will encourage increases in innovative efforts and broader public disclosures of new software designs, these controls will further the constitutional goal of patents to "promote the [p]rogress of . . . useful arts."[49] The desirability of software patents therefore turns on whether patent protections for useful software designs will encourage software developers to expend more time and effort on developing, perfecting, and publicizing those designs.[50] If the promise of patent protections seems likely to encourage these types of incremental efforts, software patent rights should be recognized because these rights will increase the number and diversity of useful software advances that are developed and made available to the public.

---

[48] Absent a license from the patent holder, a party who uses a patented software innovation in new software code will be liable for patent infringement. *See* 35 U.S.C. § 271(a) (1994) ("[W]hoever without authority makes, uses, offers to sell, or sells any patented invention, within the United States or imports into the United States any patented invention during the term of the patent therefor, infringes the patent.").

[49] U.S. CONST. art. I, § 8, cl. 8.

[50] *See* Randall M. Whitmeyer, *A Plea for Due Processes: Defining the Proper Scope of Patent Protection for Computer Software*, 85 NW. U. L. REV. 1103, 1123–24 (1991) (noting the desirability of extending patent protections to software turns on whether those protections will "encourage socially beneficial innovations" and "increase the innovative use of computers").

Whether or not the promise of patent protections will produce new levels of software development and disclosure cannot be assessed with certainty. Past studies, however, have identified a number of ways that patent protections tend to increase the number and value of publicly available inventions.[51] We can begin to assess the merit of patent protections for software advances by determining whether these previously recognized benefits of patent protections are likely to be realized by enforcing software patents. This section will assess whether software patents are likely to produce the same sorts of public benefits that are produced by patents on other types of technological advances. The next section of this article will analyze whether differences between software advances and other technological advances justify patentability standards for software that are more restrictive than for other types of useful inventions.

Software patents and related incentives to software developers may produce at least five types of public benefits: (1) encouraging inventive efforts through the promise of economic rewards to successful inventors; (2) promoting public disclosures of useful inventions through issued patents and broad scale invention marketing; (3) furthering the investment of resources in the refinement and popularization of inventions to achieve the commercialization and widespread availability of patented products and services; (4) ensuring prospective studies to identify additional or improved applications of inventions; and (5) limiting duplicative efforts to discover, perfect and improve patented inventions, thereby maximizing society's net gain from each patented invention.

A. *Encouraging Inventive Efforts*

Patents on useful inventions can further public interests by establishing incentives for the creation of increased numbers and types of inventions. Under this view, patent laws establish a

---

[51] *See* Seth A. Cohen, *To Innovate or Not to Innovate, That is the Question: The Functions, Failures, and Foibles of the Reward Function Theory of Patent Law in Relation to Computer Software Platforms*, 5 MICH. TELECOMM. TECH. L. REV. 1 (1999) (Nov. 23, 1998) <http://www.mttlr.org/volfive/cohen.html> (pointing out that rewards of patents, such as exclusive ownership of rights, also function as a benefit to the public in terms of increased technology); Whitmeyer, *supra* note 50, at 1129 (noting that the potential for a patent may entice programmers to create new software).

reward system that is aimed at encouraging inventors to invest sufficient resources in research efforts.[52] This approach to patents is based on two assumptions. As summarized by Professor Donald F. Turner:

> The basic rationale of the patent system can be simply put. The economic case rests upon two propositions: first, that we should have more invention and innovation than our economic system would provide in the absence of special inducement; and second, that the granting of a statutory monopoly to inventors for a period of years is the best method of providing such special inducement.[53]

The use of patent restrictions to constrain the use of inventions might at first seem inconsistent with the general opposition of our antitrust laws to monopolies.[54] In a strict sense, however, patents do not confer monopolies, but rather, grant exclusive and temporary control over the production and marketing of particular patented products and services.[55] These patented products and services are typically offered to consumers in competition with non-patented substitutes, meaning that a patent holder will seldom have complete monopoly control over a market.[56]

The exclusive control that a patent confers over a new invention may restrict public access to that invention. A temporary period of exclusive control over a particular invention, however, will often be a desirable price to pay for an invention that an innovator would not have pursued without the promise

---

[52] *See* Cohen, *supra* note 51, at 5–7 (stating that patents create exclusive ownership rights that enable the creator to recover the cost of invention, as well as receive accolades for their creation); Whitmeyer, *supra* note 50, at 1123–1124 (advocating that patents encourage the creation of software by rewarding inventors with exclusive rights to the programs).

[53] Donald A. Turner, *The Patent System and Competitive Policy*, 44 N.Y.U. L. REV. 450, 450–51 (1969).

[54] *See* Marina Lao, *Unilateral Refusals to Sell or License Intellectual Property and the Antitrust Duty to Deal*, 9 CORNELL J.L. & PUB. POL'Y 193, 193 (1999) (noting that intellectual property and antitrust law complement each other because they further the mutual goal of increasing innovation).

[55] *See* Scapa Dryers, Inc. v. Abney Mills, 269 F.2d 6, 13 (5th Cir. 1959) (noting that a patent would be invalid if it gave the holder an exclusive right in perpetuity).

[56] *See generally* Edmund W. Kitch, *Elementary and Persistent Errors in the Economic Analysis of Intellectual Property*, 53 VAND. L. REV. 1727, 1730–1731 (2000) (explaining that patents never provide a monopoly over the market because the patent is either very narrow in a well-developed field or broad when there is little demand for such a patent).

of some control over the invention and the resultant opportunities for economic gain which stem from such control.[57]

Without the promise of patent protections and controls over the use of their inventions, innovators might fear appropriation of their efforts by "free riders."[58] That is, innovators might fear that, once their innovations were made public, other parties might simply adopt the innovations without compensating the innovators. Allowing innovators to bear the costs of innovation while "free riders" reap the benefits may cause potential innovators to forgo product and service innovation and focus their efforts on other activities with a greater possibility of a positive personal reward.[59]

The potential threat of "free riders" may deter innovative efforts for a number of reasons. First, innovators might not want to incur large research expenses where there is no guarantee that they will have a chance to recover those expenses out of profits from exclusive opportunities to market the resulting products. Inventors will have guarantees of exclusive marketing opportunities for inventions covered by patent rights, but will have no such assurances regarding other publicly disclosed inventions. Second, innovators might not wish to risk subsidizing competitors' profits by making discoveries that the competitors could then use to improve their own products and market positions without providing appropriate compensation to the innovators. This process would simultaneously weaken the innovators while strengthening their competitors.[60]

Patent controls prevent "free rider" problems and associated deterrents to innovation by allowing innovators to either exclude

---

[57] *See* A. Samuel Oddi, *Beyond Obviousness: Invention Protection in the Twenty-First Century*, 38 AM. U. L. REV. 1097, 1102–07 (1989) (concluding that the history of the patent indicates that inventions should be encouraged by providing patents).

[58] Rebecca S. Eisenberg, *Patents and the Progress of Science: Exclusive Rights and Experimental Use*, 56 U. CHI. L. REV. 1017, 1025 (1989). Fear of this scenario—in which the non-innovators are "free riders" because they gain the benefit of the innovation or a "free ride" without compensation to the innovators—will cause many innovators to forego the risk of innovation. *See id.*

[59] *Id.*; *see also* JOHN B. CLARK, ESSENTIALS OF ECONOMIC THEORY 360 (1927); EDITH TILTON PENROSE, THE ECONOMICS OF THE INTERNATIONAL PATENT SYSTEM 26–31 (1951); SENATE SUBCOMM. ON PATENTS, TRADEMARKS, AND COPYRIGHTS, 85TH CONG., AN ECONOMIC REVIEW OF THE PATENT SYSTEM 21, 25 (Comm. Print 1958).

[60] *See* Eisenberg, *supra* note 58, at 1028 (noting that if protection of a software innovation was not offered by means of a patent, inventors would not disclose their invention for fear that a rival would copy it).

all others from the commercialization of patented discoveries or to license others to undertake this commercialization in conjunction with compensation to the innovator. By lowering concerns about "free rider" problems, patent rights should encourage innovators to pursue heightened levels of research, leading to additional inventions that would otherwise have remained undeveloped.

In addition to providing incentives for innovation, patent controls over new inventions have the added advantage of being easily administered and tailored in economic value to the importance and cost of an invention. As noted by John Stuart Mill, the temporary grant of an "exclusive privilege" under a patent is preferable to the grant of a governmental bonus to a successful innovator because the patent system avoids "discretion" of public officials regarding the allocation of such bonuses and provides a reward proportionate to the "usefulness" of a patented invention that is paid by users of the invention.[61] Potential users of an invention with little practical utility will not pay much, if anything, to the patent holder to buy or use a patented invention since the most the users will be willing to pay is the incremental value of the results achieved by the invention over and above the value of the results achieved by unpatented substitutes. By contrast, consumers will pay larger amounts for access to a patented invention that is highly useful, resulting in large sales or royalty payments to the patent holder. Scaling of rewards to inventors in relation to the utility of their inventions occurs naturally through market transactions regarding sales or licenses of patented inventions.[62] No administrator need evaluate or "price" the proper reward for a given innovation.

While a patent holder's charges for use of a patented invention may be considerable, these charges will seldom, if ever, be so large as to preclude users of the patented invention from achieving a net benefit. The charges will usually not exceed the invention's benefits to users because potential users must see a positive reason to adopt the patented invention before they will begin to use it. If the perceived benefits of using the invention

---

[61] *See* JOHN STUART MILL, *Principles of Political Economy, in* COLLECTED WORKS OF JOHN STUART MILL 1, 928 (J.M. Robson ed., Univ. of Toronto Press 1965).

[62] *See* Dan L. Burk, *Patenting Transgenic Human Embryos: A Nonuse Cost Perspective*, 30 HOUS. L. REV. 1597, 1618 (1993) (stating that market conditions may well force patent holders to price their products competitively).

are not greater than the access charges demanded by patent holders, then potential buyers or licensors will forego the invention and continue to rely on substitutes.

If the benefits of use of a new invention are greater than the charges by a patent holder for access to the invention, users of the invention, and society in general, will achieve a net benefit from use of the invention. Elevated purchase prices or royalty fees that invention users pay to a patent holder will simply shift some of the new wealth generated by use of the invention from the users to the innovator who made the new wealth possible. As noted by Jeremy Bentham, rewards stemming from the granting of "exclusive privileges" under patents are like bonuses given to innovators upon successful discoveries.[63] These bonuses are paid, in effect, out of the increased public gains and benefits achieved by the discoveries. According to Bentham, it follows that the grant of exclusive patent privileges for the purpose of improving public life is "the best proportioned, the most natural, and the least burdensome [means of] produc[ing] an infinite effect and cost[ing] nothing."[64]

As they do for other technological advances, patent rewards for innovative software are likely to encourage increased investigation and discovery of new types of useful software.[65] This is the case because the development of new software raises the types of "free rider" problems that patent laws and rewards can overcome.

Absent patent protections, the risk of appropriation of software innovations by competitors may cause potential innovators, and potential investors who might back the innovators, to divert their resources to more promising ventures. As put simply by one computer industry attorney:

---

[63] *See* Mark F. Grady & Jay I. Alexander, *Patent Law and Rent Dissipation*, VA. L. REV. 305, 310–311 (1992) (discussing temporary monopolies as a good reward for inventors).

[64] JEREMY BENTHAM, *Manual of Political Economy, in* THE WORKS OF JEREMY BENTHAM 31, 71 (1962); *see also* Grady & Alexander, *supra* note 63, at 310–314 (discussing the debate between the "reward theory" and the "prospect theory" of patents).

[65] In addition to encouraging the development of new software, broadly inclusive standards for software patents can have a positive impact on the development of new products in fields that use computer-controlled industrial processing or microprocessor-based product designs. *See United States Patent and Trademark Office, supra* note 6, at 78 (statement of Victor Siber, Senior Corporate Counsel, IBM Corporation).

> Going into the next century, the key inventions will be in information processing. . . . [Restricting patents] for software-related inventions will shift investment away from this area.
>
> The purpose of research and development in any technology is to gain an advantage over your competitor. But if your competitor can legitimately copy the fruits from your [research and development] and can create a product that can compete head-on with your product while you are still trying to build a market for the product, then you've lost.
>
> The long-term value of [research and development] in the marketplace is in the new functions implemented by software. If such new functions are protected, investment flows to the industry. If not, investment will dry up.[66]

Risks of misappropriation of new software are added to a variety of other early-stage risks that may impede the development of new software. Software patents can play a critical role in overcoming these risks and producing the investment that is needed to back new software development:

> The software industry is different than some other industries. It has a very front-end loaded investment. You have to invest all of your risk capital before you know whether or not your product is going to be competitive, before you know whether or not anyone will buy your product. So you have a lot of risk and you have a lot of uncertainty.
>
> If you add on that additional risk and uncertainty associated with not knowing whether or not you'll be able to protect your new and innovative product from a major competitor, you may not have any investment at all. The market risks are very high, and if competitors can take a utilitarian function which you spent a lot of time and effort designing . . . and use [that function] against you without having the same [research and development] expenses, you would not be able to support these investments.[67]

"Free rider" problems can impair the development of new software because efforts to develop complex software can involve significant outlays for design studies which may not produce accompanying returns for the developer if others can freely use

---

[66] *Id.*

[67] *United States Patent and Trademark Office, supra* note 6, at 65 (statement of Tom Cronan, General Counsel and Secretary of Taligent).

the design study results.[68] Several types of studies related to software development may entail significant costs. Studies of information processing bottlenecks and inefficiencies are typically important in defining the necessary parameters of a software advance. Trials of several new information processing approaches may be necessary before an advantageous advance is identified. Substantial program coding efforts may be necessary before a software product developer can identify successful means to achieve desirable performance in a software product. Additional studies may be needed to supplement an operative software product with an attractive and effective user interface.[69] All of these features of software development heighten the costs of producing successful products and increase the risks of "free rider" misappropriation of research products absent patent controls.

"Free rider" problems concerning software development may be particularly serious because of the ease with which software advances can be copied and utilized by parties other than the original developers.[70] Once software incorporating a new information processing advance is made public, the software will often reveal enough about the advance to make it possible to replicate the advance in software created by other parties. Even

---

[68] The high costs of design studies and tests that are often necessary to produce a workable software product stem in part from the complex nature of the interactions between functional elements in large software products. While conceptualization of the desired functioning of a software product or product module may be relatively simple, the implementation of the concept in functioning programming code that works accurately and reliably in conjunction with the rest of a software product may be very difficult:

> Despite the ease under which someone can . . . [describe information processing steps to be undertaken by a computer], we still live under real world constraints. Once a design choice is made, it is very expensive in time and effort to change it. Worst, because most programs have interactions that cover every part, a change to one part can cause unexpected and even undesirable side effects in unknown and unexpected places.

United States Patent and Trademark Office, supra note 1, at 4 (statement of Paul Robinson, Chief Programmer, Tansin A. Darcos & Company).

[69] *See* Jane M. Rolling, *No Protection, No Progress for Graphical User Interfaces*, 2 MARQ. INTELL. PROP. L. REV. 157, 160 (1998) (discussing the importance of user interface).

[70] *See* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protections of Computer Programs*, 94 COLUM. L. REV. 2308, 2332 (1994) (stating that the know-how in software products is easily copied, undermining the developers opportunity to profit and creating disincentives for investing in software development).

if the production of these functionally similar follow-on products involves writing some new programming code to implement the new advance, this type of reprogramming based on the model provided by the innovator's software will often be far easier to accomplish than the original development and coding of the innovator's product.[71] The relative ease with which this can be accomplished—coupled with the lack of any obligation of the second and subsequent users of the advance to pay a licensing or use fee to the innovator that developed the advance—would give these later users a substantial economic advantage over the original developer. Fear that these later "free riders" will adopt and benefit from costly software advances will be a real concern resulting in corresponding deterrents to software development.

This sort of "free rider" misappropriation is precisely the type of problem that patent protections and rewards are designed to prevent. The importance of patent protections in overcoming similar "free rider" problems in other fields suggests that patent protections for new software designs are justified to diminish "free rider" concerns on the part of software innovators and to encourage heightened levels of complex software development.

Of course, software innovators may gain some assurances of exclusive opportunities to commercialize new products through intellectual property protections other than patent rights. Copyrights, in particular, will prevent a party from merely copying an initial innovator's programming code and thereby appropriating product improvements resulting from the innovator's efforts.

While copyright laws provide partial protections for software innovators, these protections are less comprehensive than patent controls in several important respects. First, copyrights may not protect the full value of an innovator's efforts. Many useful features of a new software product may be discoverable and reusable by programmers without copying the software code involved. Absent such copying, the reuse would probably not support a copyright infringement claim.[72] The functional

---

[71] *See* Rolling, *supra* note 69, at 178–79 (stating that the cost of duplicating a program is far cheaper than developing a computer program).

[72] Functional features of an innovative software product may be discoverable through mere inspection of the operation of the product. Alternatively, these features may be discoverable through reverse engineering of innovative software.

attributes of innovative software can, however, be protected by patents against reuse in additional software products with similar functionality. Second, copyright claims may entail insurmountable evidentiary barriers. In order to pursue a claim of copyright infringement, a software innovator will need to establish that an asserted infringer has copied protected portions of the innovator's product.[73] Usually, this will require a showing that the asserted infringer copied part or all of the innovator's programming code.[74] Tracing this copying may be difficult. By contrast, a patent holder merely needs to show that a second party has incorporated information processing steps in the second party's software product which are the same as or equivalent to the ones protected by the first party's patent.[75] The fact that the second party may have independently developed these steps without copying the first party's software code, or without even having access to that code, will be irrelevant since independent development and use of a patented invention is not a defense to patent infringement. In light of these weaknesses, copyright protections are not a full substitute for the protections against "free rider" problems provided by software patents.

B. Promoting Invention Disclosures

Whatever impact patents have in diminishing "free rider" concerns and increasing the scope of inventive efforts and discoveries, patent protections can also provide additional public benefits by encouraging parties who have made valuable discoveries to disclose those discoveries, thereby making them broadly available to the public.[76] In exchange for the public

---

Reuse of these functional features will not constitute infringement of the copyright on the programming code contained in the innovative software product.

[73] *See* Daniel N. Christus et al., *Practicing Law in the Americas: The New Hemispheric Reality: Intellectual Property in the Americas*, 13 AM. U. INT'L L. REV. 1095, 1125 (1998) (contrasting copyrights, which prevent exact copying of an original work, with patents, which provide more protection for an inventor).

[74] *See id.*

[75] *See id.*

[76] As noted by one leading commentator:

> [T]he property rights (and thereby the incentives) made available under patent laws are tied to compliance with a clear public policy promoting disclosure and public availability of innovative techniques and inventions .... The public record [created by an issued patent] provides an information base for use in further innovation by other parties. Thus,

disclosure of an invention in a published patent, an inventor is given the reward of monopoly control over the invention for the life of the patent. This type of disclosure and the rewards necessary to encourage it serve the public interest in at least four ways: (1) disclosure will help to make the availability of the invention, under sale or license from the patent holder, known to those who may have an immediate use for it; (2) disclosure will permit persons other than the original inventor to extend the invention through further inventive efforts. Resulting improvements may produce useful products or processes in the original field of the invention or in other settings; (3) disclosure facilitates understanding and free use of the invention when the patent expires; and (4) descriptions of an invention in a published patent may reveal new design techniques or technological information that are not protected by the patent. Patent developers and others can use these techniques or information immediately upon issuance of the patent.

Patent controls over the use of inventions can overcome several barriers to disclosures that will arise in the absence of these controls. In settings where innovative software can be used effectively in secret—as would be the case, e.g., with computer-aided manufacturing software that was used to make products without disclosing the manufacturing software—a party may seek to commercialize the software without making any disclosures in order to withhold knowledge of the software from the party's competitors.[77] Using physical security measures and trade secret protections, a software developer may be able to pursue this secret commercialization and gain a significant return on his or her software without public disclosure.[78]

> while the system establishes restrictive rights for the inventor, it also promotes the flow of scientific and technological data and development, thereby perpetuating a process of innovation among a community of scholars using shared information.

RAYMOND T. NIMMER, THE LAW OF COMPUTER TECHNOLOGY ¶ 2.02[1] (2d ed. 1996); *see also* Wood v. Underhill, 46 U.S. 1, 5 (1847) (holding that where an inventor has not described and disclosed an invention with sufficient particularity, the inventor is not entitled to a patent); Donald S. Chisum, Comment: *Anticipation, Obviousness, Enablement: An Eternal Golden Braid*, 15 AM. INTEL. PROP. L. ASS'N Q.J. 57 (1987); ROBERT PATRICK MERGES, PATENT LAW AND POLICY 657–58 (2d ed. 1997).

[77] *See* Robert G. Bone, *A New Look at Trade Secret Law: Doctrine in Search of Justification*, 86 CAL. L. REV. 241, 262–63 (1998) (discussing the possible chilling effects of public disclosure and reasons for commercializing in secret).

[78] *See id.*

Such secret commercialization will come at the price of lost public access to the software involved. This may, in turn, lead to inefficient use of the software under the constraints of confidentiality controls.[79] It may also produce duplicative efforts by others to develop the same software.

Patent rights for innovative software should be valuable means to overcome disclosure barriers. Patent protections will provide software developers and companies with means to maintain exclusive control over innovative features of new software while still disclosing the software and related business practices. The control provided by patent rights can overcome the incentives for secret commercialization of useful software by giving software developers a means to realize the commercial value of their discoveries through public disclosures without fear of appropriation of the discoveries by competitors.[80] Patent protections will also assist firms in gaining the full use of software advances without the awkwardness and often wasteful expense of secrecy measures or use restrictions aimed at avoiding software disclosures.[81]

The reassurances provided by software patents may also encourage additional types of software disclosures beyond those in patent applications. Voluntary disclosures of new software advances will be encouraged because individuals and companies know that they can make these disclosures without losing control over the disclosed advances. Provided that they file a timely patent application covering the advances, disclosure in public presentations or distributed writings will not undercut the innovators' patent rights.[82] Voluntary disclosures of new

---

[79] *See id.*

[80] *See United States Patent and Trademark Office, supra* note 1, at 29 (statement of A. Jason Mirabito, Boston Patent Law Association) (attributing the more frequent publication of biotechnology innovations than software innovations to the greater reliance on patent protections in the biotechnology industry and the tendency of software innovators to rely on commercialization of their software in confidential settings through trade secret licenses).

[81] *See* Syrowik, *supra* note 1, at 117 (stating that patents are typically preferable to trade secrets because businesses find it increasingly difficult to maintain secrecy).

[82] In general, under United States law, a timely patent application will need to be filed within one year after the first disclosure of an advance. *See* 35 U.S.C. § 102(b) (1994). Under the laws of many foreign countries, a patent application must be filed before any public disclosure of an advance. *See, e.g.*, Donald R. Palladino, *The Publication Bar: How Disclosing an Invention to Others can Jeopardize Potential Patent Rights*, 37 DUQ. L. REV. 353, 354 (1999). Even in countries that

software advances in technical publications or published product descriptions may be both more extensive (for marketing or public relations reasons) and more rapid than the disclosures which eventually result from the issuance of a patent. The impact of software patents on voluntary disclosure practices at the AT&T Corporation was described by one of its attorneys as follows:

> One of the functions . . . patents [serve] is to disclose [new technological advances] to the public . . . . Patents themselves of course contain disclosure, but also in an organization like mine . . . we encourage publication of technical ideas, in fact last year we published some forty-four hundred technical articles. Many of these would not have been published if we could not also have concurrently filed patent applications so that the publication of the technical papers would not compromise the value of our inventions included in the disclosures.[83]

Patent rights may also expand the range of persons who are informed about innovative software. These rights will encourage broad dissemination of information about a new software advance by giving the holder of the patent a direct financial interest in the full range of potential uses of the new software.[84] Patent rights give innovators the ability to restrict and license the use of new software in fields and applications outside of the setting where the software was originally developed. The potential economic rewards associated with this licensing opportunity will encourage innovators to publicize the functional characteristics and benefits of new software in all of the fields and applications where that software appears likely to be useful.[85] By pursuing this type of publicity at a reasonable level,

---

apply this stricter rule, the reassurances provided by patent rights encourage public discussions and disclosures of new discoveries at the time a patent application is filed. *See* 35 U.S.C. § 154(a) (1994). This achieves disclosure of the inventions involved without the delay entailed in processing the patent application.

[83] *United States Patent and Trademark Office, supra* note 6, at 7 (statement of William Ryan, AT&T attorney); *see also United States Patent and Trademark Office, supra* note 1, at 23 (statement of Richard Jordan, Patent Counsel, Thinking Machines Corporation) (describing the impact of software patents in promoting the publication by company employees of descriptions of software advance; the company did not restrict these publications because the concern was able to protect its advances adequately through software patents).

[84] *See* Cohen, *supra* note 51, at 3 (stating that the reward function of the patent system creates incentives for innovation and disclosure).

[85] *See* Richard D. Nelson & Roberto Mazzoleni, *Economic Theories About the Costs and Benefits of Patents*, 32 J. ECON. ISSUES 1031 (1998) (last visited Oct. 29,

a patent holder will be advancing his or her self-interest since the publicity will be likely to result in incremental product sales or licensing revenues. Absent patent rights, however, there would be little incentive to disseminate information about a new software advance to parties in other fields since the party making the advance would have little, if any, means to benefit from this publicity.[86]

The benefits achieved through broader disclosures of software inventions relate not only to the public's use of the software being disclosed, but also to the avoidance of duplicate research to develop equivalent software.[87] Such duplicative software development is, in part, a detrimental offshoot of past uncertainty in software patent standards. Until recently, restrictive views on software patents held by the PTO and certain courts caused relatively few software patents to issue and corresponding gaps to develop in the record of software development reflected in issued patents.[88] The result, according to one computer science professor, was an unnecessary duplication of programming effort due to the lack of accumulated software innovation disclosures:

> Much of computer science I see . . . consists of reinventing wheels. A large amount of that is because people don't check prior art and a large amount of that is because there's no good prior art collections t[o] [ch]eck, and I think . . . that this is one of the problems that has been caused by the two decades of the

---

2001) <http://www.nap.edu/readingroom/books/property/3.html#chap3> (suggesting that patents facilitate further development and use of the patented product).

[86] *See id.* For example, consider a software developer who creates a new information processing system for storing insurance company records based on the risk characteristics of various insurance policy holders. Information-storage systems indexed by customer risk characteristics may also be useful outside of the insurance field. For example, in programs used by retailers to identify consumer needs and likely product or service preferences. Once a software innovator, who develops an effective means to correlate record storage with consumer risk, is granted a patent on this innovation, the innovator will have a stake in all of the potentially productive applications of this invention. He or she will therefore see that it is publicized and incorporated in software products in all of the fields where the software has foreseeable applications. With these incentives for broad disclosure and advocacy of additional applications, there will be an increased chance that an advance which was initially developed for a narrow field like insurance records storage will be disclosed to and used by a broader range of persons who might gain from the advance.

[87] *See id.*

[88] *See United States Patent and Trademark Office, supra* note 6, at 30 (statement of Lee Hollaar, Professor of Computer Science, University of Utah).

> Patent Office having at best ambivalent attitudes toward the patentability of computer software and not using the patent system to draw the trade secrets and the other art into the printed publications of U.S. patents.[89]

This type of duplication—and the corresponding waste of some of the net value of the software innovations involved—can be avoided through greater use of software patents and greater reliance on the records of software development that an expanded number of patents will produce.

C. *Providing Incentives for Product Refinement and Popularization*

Many inventions involve workable but rudimentary designs of new devices or processes that require considerable further refinement to transform them into useful and marketable products.[90] Extensive engineering adjustments may be needed to make an invention work well enough to form the basis for marketable products. Substantial post-invention industrial engineering may also be needed to initiate mass production and distribution of products based on a new invention. For example, mass production of a product may only be possible following costly production engineering studies to set up manufacturing operations. Extensive product introduction advertising may also be needed to familiarize consumers with the new features of the product. These product perfection and popularization efforts—distinct from the research activities necessary to produce an invention—will be referred to here as product introduction activities.

Patent protections can serve a valuable societal function by encouraging patent holders to invest resources in product introduction activities.[91] Absent the guarantees of exclusive

---

[89] *Id.*

[90] *See* Nelson & Mazzoleni, *supra* note 85 (discussing how patenting of inventions occurs early in the process, and therefore much work is often needed to make the invention viable).

[91] This impact of patent protections was first analyzed in depth by Joseph A. Schumpeter. *See* JOSEPH A. SCHUMPETER, THE THEORY OF ECONOMIC DEVELOPMENT 67–74 (Redvers Opie trans., 1983); JOSEPH A. SCHUMPETER, CAPITALISM, SOCIALISM, AND DEMOCRACY 81–110 (3d ed. 1950); JOSEPH A. SCHUMPETER, BUSINESS CYCLES 84–192 (2d ed. 1964) (1939). Schumpeter emphasizes the importance of post-invention product innovation as a means to bring about revolutionary changes in an economic system. Patents, in Schumpeter's view, are means to attract investment and competition in the development and marketing

product marketing that flow from patent ownership, parties who are considering the commercialization of new products will hesitate to invest substantial engineering and marketing resources in producing and promoting products that other providers might then produce and market with lower costs.[92] The production costs of second and subsequent producers will be lower if they can gain some of the benefit of the first manufacturer's product introduction activities. This will be the case if later product introducers can gain from and avoid repeating some of the first manufacturer's product perfection efforts or manufacturing start-up activities. These kinds of benefits can sometimes be realized by studying the first manufacturer's products and methods of manufacturing (to the extent that these methods are publicly known). Similar benefits can also be obtained by introducing new products as substitutes for the first manufacturer's products in ways that capitalize on consumers' familiarity with the products based on the first manufacturer's product introduction advertising.

Where these practices can be freely undertaken by second and subsequent product introducers, they can create "free rider" problems. Absent patent controls, a potential producer and marketer of a given type of product may be hesitant to be the first to work out the manufacturing problems for a new type of product and to expend the advertising and other marketing resources necessary to inform the public about the nature and merit of the new product.[93]

---

of new products following the discovery of a new technology. *See* JOSEPH A. SCHUMPETER, THE THEORY OF ECONOMIC DEVELOPMENT 67–74 (1983). He argued that radical market changes often resulted where new firms arise to exploit new innovations, driving out old firms that provide obsolete goods and services. *See* JOSEPH A. SCHUMPETER, CAPITALISM, SOCIALISM, AND DEMOCRACY 82 (3rd ed. 1950). The prospect of extraordinary returns, however, from the production and marketing of products based on a new technology was necessary in order to induce investment and to lure productive resources away from other uses. Hence, control granted through a patent covering the implementation of a new technology could increase rather than restrict the availability of that technology and related products, by facilitating the introduction of the products by innovating firms. *See id.*

[92] *See* Syrowik, *supra* note 81, at 117 (discussing how investors who sponsor start up companies seek some certainty as to protecting their investment).

[93] *See* Paul E. Schaafsma, *An Economic Overview of Patents*, 79 J. Pat. & Trademark Off. Soc'y 241, 243 (1997) (noting the ability of competitors to avoid the high risks of introducing new products by only producing and selling substitutes for other businesses' already popular products, for which patent protection is not available); *see also* Marsha J. Ferziger, *Monopolies on Addiction: Should*

Once an initial product introducer has designed a workable product and created an associated market demand, later suppliers of the same product will be able to produce the product in competition with the first firm,[94] but will not have to bear all of the engineering costs and product introduction outlays incurred by the original inventor. The later producers would therefore be able to profitably market their products at lower prices and win a substantial portion of the market for their products at the expense of product sales by the product innovator. The second and subsequent producers are free riders with respect to product engineering, manufacturing start-up, and product introduction expenditures of the first manufacturer that also benefit the subsequent producers. Absent patent restrictions on their manufacturing and marketing of substitute products, second and subsequent introducers of versions of an innovative product will not need to pay any share of the expense of the product introduction activities from which they benefit.

The foreseeability of these sorts of "free rider" effects, and fears that these effects will subsidize competitors, may cause firms that are capable of incorporating new inventions into products to forgo such changes and to invest their resources elsewhere. The result will be that useful inventions may be discovered, but not carried forward into generally available products.

"Free rider" effects related to product perfection and popularization are likely to cause serious problems concerning software innovations. Once an individual or a company

---

Recreational Drugs be Patentable, 1994 U. CHI. LEGAL F. 471, 475 (1994) (stating that without patent protection there would be little economic incentive to create and research).

[94] Of course, the design and production of products that are "clones" of an innovative product takes some time. During the delay between introduction of the first version of an innovative product and the introduction of its clones, the original market entrant will have a temporary monopoly over sales of the product. The value of this temporary monopoly will depend on the functional superiority of the new product over substitutes, the time necessary to bring duplicate products to market and the willingness of consumers to wait for similar, and less costly, products from other producers. In some settings—for example, operating system software for personal computers—the difficulties and delays in producing work-alike products give initial producers large, and sometimes insurmountable, marketing advantages. *See* Dan L. Burk, *Muddy Rules for Cyberspace*, 21 CARDOZO L. REV. 121, 166 (1999) (explaining that the most profitable time for an invention may be the period between when the invention actually reaches the market and the time when clones of the invention reach the market).

identifies a potentially useful software advance, significant product development and popularization expenditures may be needed before related products can be widely used by the public.[95] Resources are unlikely to be expended on these tasks if the parties involved think that their efforts may benefit not only their own companies, but also benefit competitors who can wait for the initial party's product designs and product introduction efforts to succeed and then market knock off products without bearing all of the associated product design and popularization costs.

Significant product development costs can be incurred in the perfection of software products due to the complexity of large-scale programming projects. The development of a useful software product that incorporates a new software design feature may involve large expenditures of resources to integrate the new advance with the other aspects of the software product and produce reliable program functionality and results.[96] Software development projects—even those involving modifications to a previously stable, working product—are notorious for involving large and unpredictable programming, testing, and debugging costs.[97] The size of these costs suggests that software vendors may be deterred from initiating software improvement efforts if they cannot be sure of gaining most of the increased product sales and profits resulting from those efforts.

Public education concerning a new software product may also add substantial costs to the product introduction expenses that must be borne by the first innovator to market such a product. Public education about a new product will include advertising and product demonstrations to illustrate and explain

---

[95] *See* Simone A. Rose, *Patent "Monopolyphobia": A Means of Extinguishing the Fountainhead?*, 49 CASE W. RES. L. REV. 509, 520–521 (1998) (explaining how an invention, which may be superior to other products on the market, will not be purchased by consumers unless the inventor is able to spend money to promote and educate consumers about the product).

[96] New software development is often an extremely expensive activity. Among the developers of widely marketed packaged software, an average of 17 percent of company revenues flows into continuous research and development efforts. *United States Patent and Trademark Office*, *supra* note 6, at 6–7 (statement of William Ryan, AT&T attorney).

[97] *See* A. Samuel Oddi, *An Uneasier Case for Copyright Than for Patent Protection of Computer Programs*, 72 NEB. L. REV. 351, 425 n.304 (1993) (explaining that debugging costs associated with software programs are difficult to estimate because of the number of variables involved with software programs).

the benefits of the new product. The costs of these sorts of efforts are transaction costs that must be incurred to inform potential software users about the merit of the new product and its innovative features. Parties marketing similar products after the initial product introduction by the software innovator need not expend these same sorts of resources since the public will already be somewhat familiar with the relevant product advance and its practical advantages. Absent some exclusivity in marketing a new type of software product (at least for some time), an innovator will be hesitant to invest significant resources in public education to promote the initial marketing of that product.

Given the scope of these costs, patent reassurances to companies developing new software products may be critical in convincing those companies to commit resources to new product development projects. The same reassurances may be needed to convince venture capitalists and other investors to make similar decisions to back start-up companies in their development and marketing of new software products.[98] Software patents can influence investment decisions because "investors seeking to sponsor a start-up organization or a new enterprise within a larger company would like to have some certitude about what it is that they can hope to have some protection for and . . . how their investments can be protected."[99]

A patent attorney described this type of impact of software patents on one of his clients:

> I have a client that's a small software company on the West Coast, initially financed through the founder's own resources. This [company's product] is a utility type of software, improving hardware performance and reliability. They filed a patent application. A hardware company that they were working with

---

[98] *See* Jonathan M. Barnett, *Cultivating the Genetic Commons: Imperfect Patent Protection and the Network Model of Innovation*, 37 SAN DIEGO L. REV. 987, 1001–02 (2000) (explaining how patent protection fosters investment by assuring that a "free rider" will not be able to capitalize on the invention and prevent the original inventor and investors from recovering their investment capital).

[99] *United States Patent and Trademark Office, supra* note 6, at 7 (statement of William Ryan, AT&T attorney) *See also id.* at 9–10 (statement of Richard LeFaivre, attorney for Apple Computer) (noting that Apple Computer and other large software producers regularly consider the patent potential of alternative development projects before embarking on those projects; projects lacking potential patent protection are disfavored because of the risks that innovative products resulting from the projects will be appropriated by competitors).

> decided to flex its muscles a bit and threatened to design their own product, notwithstanding the patent application. However, once we had an indication of reasonable allowable claims we were able to negotiate them back into the fold.
>
> A few months later, despite the success of the product . . . , marketing expenses were just eating up the company's cash.
>
> The company went to look for investors. Every single investor refused to get actively involved until knowing that there would be strong patent protection, because the one thing that makes software unique is how easy it is to copy. And I'm not using copy necessarily in the copyright sense, but analyzing it and taking what's there.
>
> [After obtaining a patent], our client is at this point closing the financing which was the difference between life and death for the company.[100]

Patent rights held by an innovator can also provide the basis for technology transfers to other parties who will develop or apply a new technology. This use of software patents is particularly important in university environments where institutions seek to license outsiders to use software developed by university personnel. One attorney described the key role played by the licensing of university-held patents as follows:

> Universities will generally not be able to license their technology unless they have a chance of protecting it. They are not known to be litigious: it is out of respect for the patent system and access to future technology generally that a licensee signs up. [My law firm has] seen a number of instances where software developed at universities was licensed by the very

---

[100] *United States Patent and Trademark Office, supra* note 6, at 35–36 (statement of Steven Henry, Attorney, Wolf, Greenfield & Sacks, P.C.). While trade secret rights may protect software advances that can be commercialized in secret, these sorts of protections may not be adequate to attract investor support for company expansion. There are several reasons why potential investors in a company developing a software advance may believe software patent protections are superior to similar trade secret protections:

> [T]here's often a choice between whether to keep processes secret or obtain a patent on it. I find investors like patents much better, for two reasons. One is, they don't like dealing with trade secrets because they have to sign a confidentiality agreement and a lot of investors won't do that. The second reason is, and I think even more compelling, is that the investors are afraid that the trade secrets will have a short lifetime. They can easily be lost. They can be lost in an instant by an inadvertent publication.

United States Patent and Trademark Office, supra note 6, at 41–42 (statement of Robeert Yoches, Attorney, Finnegan, Henderson, Farabow, Garrett & Dunner).

> developers who knew the potential, went out, formed their own companies, and that was a revenue stream that was formed back to the universities and that revenue stream is very important.[101]

Patents can form the basis for bringing together partners who will contribute differing expertise or resources to the development and marketing of a new software product. An innovator can contribute patent rights to a new partnership formed with other specialists and the partnership can then use these rights to ensure exclusive marketing of the products or services it offers. Such partnerships are particularly common in the software field since the production of a successful product often requires a mix of programming and application domain expertise that is uncommon in one company or organization.[102] One computer industry attorney described the impact of patents as the linchpins in forming productive partnerships in the following terms:

> [Patents] provide a vehicle for developing of the ubiquitous alliances that are present in the [computer] software and hardware industries. They provide a medium, in fact, for people to come together and exchange value so that they can work together to get a cooperative result. Often this helps people and companies get into new markets and establish businesses that would not otherwise exist.[103]

Among the partners in a venture to commercialize a new software-based product, software patents may be a means to strengthen the hand of software developers and provide them with a greater fraction of the overall profits from the product. The impact of software patents in clarifying the value of software assets and, by extension, the value of programmers' contributions to innovative software may significantly change our notions of power and commerce concerning software products:

---

[101] *See id.* at 36 (statement of Steven Henry, Attorney, Wolf, Greenfield & Sacks, P.C.).

[102] *Partnering for Competitive Advantage; Strategic Software Relationships; Special PDA Engineering Advertising Supplement*, 11 COMPUTER AIDED ENG'G 10, S9 (1992) (stating that establishing partnerships has become an effective way to accelerate technological development, enhance productivity, promote teamwork, increase the flow of innovation and promote the sharing of information).

[103] *United States Patent and Trademark Office, supra* note 6, at 7 (statement of William Ryan, AT&T attorney).

> In today's global highly competitive marketplace, some believe that we are witnessing a fundamental shift in business history. They are, we say, progressing from managerial capitalism to intellectual capitalism. They believe that the importance of intellectual capital will ultimately cause a dramatic shift in the wealth of the world from material resources to those who control ideas and information that is intellectual property.
>
> A fundamental feature of the patent system is that it establishes a basis for this intellectual effort to be regarded as an asset and to be traded in the marketplace. Thus, an effective patent system that promotes creativity by providing a beneficial and stimulating environment for inventors is essential for the information age.[104]

D. *Encouraging Efficient "Prospecting" for Applications and Improvements*

Patent controls over the use of an invention can have further beneficial impacts on searching or "prospecting" for additional applications of the invention and improvements to the invention. Once an innovator gains a patent covering an advance, the innovator (or a successor who holds the patent) will generally seek to maximize the value of the patent by increasing use of the patented invention. The use of a patented invention can be increased in at least two ways: by finding new applications for the patented invention and by seeking improvements on the existing invention that will make it more useful or effective in its present applications and thereby increase the frequency with which it is used. Because patent holders will generally realize greater product sales profits or licensing royalties as patented inventions are used more often, patentees are encouraged to invest reasonable amounts of resources in "prospecting" for new types of applications and improvements of their patented inventions.[105]

This interpretation of patent incentives is premised on the view that inventive processes are comprised of two distinct stages: an initial discovery phase in which a new invention is

---

[104] *United States Patent and Trademark Office, supra* note 1, at 26 (statement of Ron Reiling, Digital Equipment Corporation).

[105] *See* Cohen, *supra* note 51, at 19 ("When patent rights are claimed in a particular technology, and that technology is adopted by consumers, there is increased prospecting activity in that technological field.") (citation omitted).

identified and a second application development and improvement phase in which useful applications and improvements of the invention are sought.[106]

Patent standards can be used to influence and regulate the efforts of multiple competing inventors in these two stages of invention development. By withholding patent rights and associated product controls until a clearly useful device or process design is in hand, patent standards ensure that the first of these inventive stages is an unconstrained, competitive free for all, with potential innovators encouraged to race diligently to be the first to make a discovery and thereby gain the associated patent rewards.

Once an invention is made and a patent is obtained, the second stage of searching for applications and improvements of the invention may be best conducted or coordinated by one party who has a clear economic interest in maximizing the value of the original invention and who will devote reasonable resources to the search for practical applications and improvements of the invention. This is accomplished under present patent standards by recognizing the exclusive rights of a patent holder at an early point of invention development when at least one version of an invention (not necessarily the best) has been conceived and shown to have at least one useful application (not necessarily the most important).[107]

By attaching patent rights at this early point, later efforts to apply and improve the invention can be managed by the patent holder, who has a clear economic interest in pursuing additional applications and improvements which will expand the commercialization and licensing opportunities associated with the patent. At the same time, the patent holder will be concerned that expenditures on searches for invention applications and improvements be limited to reasonable amounts such that the cost of these search efforts does not exceed the new

---

[106] *See* Marc B. Hershovitz, *Unhitching the Trailer Clause: The Rights of Inventive Employees and Their Employers*, 3 J. INTELL. PROP. L. 187, 189 (1995) ("There are two distinct parts to the inventive process: (1) conceptualization and (2) reduction to practice.") (citation omitted); *see also* Edward W. Kitch, *The Nature and Future of the Patent System*, 20 J.L. & Econ. 265, 272–73 (1977) (describing the development of inventions in two phases, one leading up to the application for a patent and another involving subsequent efforts to transform the invention into commercially successful products).

[107] *See supra* note 105 and accompanying text.

utility and commercialization value which the expanded applications and improvements are likely to achieve. Finally, because the patent holder can be confident that commercialization of new applications and improvements of the patented invention will require the patent holder's consent and compensation, the patent holder's efforts will not be deterred by fears of "free rider" appropriation of newly discovered applications and improvements.

Seen this way, the patent system serves parallel functions to those served by prospecting laws governing mining claims.[108] Both mineral prospecting laws and patent laws "avoid duplication of effort, create an incentive to invest in development, lower the cost of contracting for complementary resources by reducing the need for secrecy, and lower the claimant's cost of maintaining control over the valuable discovered resource."[109] The proper scope of patent protections under this approach is the range of further innovations a patent holder can reasonably be expected to prospect, e.g., pursue effectively, given the nature of the patentee's original discovery. Rather than defining rights concerning a claim with physical boundaries, as mining law does, a patent defines a conceptual domain of ideas about useful methods or devices and gives the patent holder exclusive control of further application development and improvement activity within that domain.

Incentives for application prospecting may be particularly important in connection with innovative software. Information processing advances implemented in software can have many applications that are unclear when first developed in a narrow application context. Expanding from its first narrow application to its full range of useful applications may require considerable studies and reprogramming.[110]

---

[108] *See generally* Kitch, *supra* note 106, at 271–75 (arguing that the patent system serves "to increase the output from resources used for technological innovation" and comparing the patent system to the mineral claim system in the the 19th Century); *see also* Cohen, *supra* note 51 (analyzing Edmund Kitch's analogy between the patent system and the prospecting laws governing mining claims).

[109] Grady & Alexander, *supra* note 63, at 314; *see also* Cohen, *supra* note 51 ("To borrow Kitch's mineral analogy, when a prospector located mineralizations in a certain area, the usual result was increased prospect activity in that particular area.").

[110] *See* Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Sofware Industry*, 89 CALIF. L. REV. 1, 23 (2001) ("[P]atent law anticipates and even depends on one party improving another party's invention").

Patent incentives encouraging software developers to search for further applications of the developer's patented software can promote two types of application prospecting. First, additional applications or improvements of patented software might be sought in the same field in which the software was originally developed. [111] A software innovator (or the assignee of the innovator's patent) will often have substantial expertise about the field of the innovation and therefore be in a good position to identify additional software applications in that field. Patent rights assuring a software innovator of exclusive control over new uses and applications of patented software will encourage patent holders or assignees to conduct these studies. A patent holder will be encouraged—in a way that non-patent holding innovator concerned with free riders would not be encouraged—to complete these studies in order to identify product applications and related business prospects that will increase the use of the underlying invention and maximize the value of the corresponding patent.

Second, in identifying the full range of useful applications of innovative software, cross-domain searching for new applications may also be important. A given type of software that is first implemented in one field may have further applications in several other fields. Persons with computer programming expertise or substantive expertise in the first field may have little reason to know of the potential importance of the software in additional fields. For example, an individual who develops a highly specialized software application such as software for controlling blood analysis devices, may lack the technical expertise to recognize that the new information processing method that the individual has recognized as a means for conducting blood analyses is also useful in monitoring steel welding processes.

It is unlikely that software developed and applied in one field will be further applied in a second field unless publicity about the advance in the first field reaches experts in the second field. Those experts must then conduct studies that confirm the usefulness in the new field of the software advances developed in

---

[111] *See* Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644, 2650 (1994) (stating that the next generation of technological innovation within software will improve upon usability of existing applications).

the first setting. These publicity and additional research efforts may be particularly complex and costly for software innovators because these efforts may extend to a wide variety of fields and contexts beyond the setting where a software advance is made. Patent rights will be a useful means to encourage reasonably scaled efforts to extend software innovations outside their initial fields of use and thereby ensure that more of the functional benefits of those innovations are brought to the public.

E. Avoiding Duplication of Inventive Efforts

Patent rights may achieve additional public benefits by discouraging the wasteful duplication of efforts to perfect or improve patented inventions.[112] Because parties seeking to make, use, or sell a patented invention (in either its original state or an improved form) need the patent holder's permission to do so,[113] a patent holder can control and regulate subsequent efforts to implement and improve the patented invention. The patent holder can ensure that a given line of product research or development is undertaken only once by parties who are authorized to do so by the patent holder, rather than being duplicated in a wasteful manner by researchers working in parallel.

By limiting duplicative efforts to improve and implement a patented invention, a patent holder can maximize the total societal gain or "monopoly rent" associated with the invention. The monopoly rent for a patented invention is the value of the increased utility associated with the innovation, less the cost of developing the invention.[114] Put another way, the monopoly rent is society's net gain from the invention. Societal gain from an invention will be maximized if the monopoly rent associated with that invention is not "dissipated" through wasteful efforts that involve greater than necessary expenditures to develop or improve the invention.[115]

---

[112] *See generally* Grady & Alexander, *supra* note 63, at 308; Kitch, *supra* note 106, at 278 (stating that once a patent is filed people will not waste effort in reproducing something already patented).

[113] *See* Grady & Alexander, *supra* note 63, at 307 (stating generally that broad protection of patent rights would allow all development opportunities to be controlled by the patent holder).

[114] *See id.* at 308 (explaining both the theory and societal benefits behind rent dissipation and how an inventor receives monopoly rights).

[115] *See id.*

The monopoly rent from a patented invention can be dissipated (and the net value of the invention to society reduced) several ways. Duplicate efforts by multiple innovators may be undertaken in races to discover inventions. That is, two or more parties may undertake parallel research leading to a single invention. The efforts of the second and subsequent inventors in such races are wasted insofar as the efforts of the first innovator alone would have been sufficient.

Patent rights tend to limit this type of monopoly rent dissipation. For example, present patent standards permit inventors to obtain patents and thereby disclose their inventions at early stages of development, e.g., when one useful application has been realized, even if efforts to refine the invention or related products are still ongoing.[116] These early stage disclosures tend to discourage the continuation of duplicative efforts by other researchers who may be undertaking parallel development efforts.[117] Researchers will tend to discontinue parallel efforts at this point because they realize that the patent will preclude them from freely using the products of their research. The researchers will be encouraged to shift from duplication of the already completed product development to instead study how to use the patented invention under a license from the patentee or how to produce non-infringing substitutes for the patented invention.

A different form of post-invention rent dissipation may occur where parties undertake duplicate efforts to improve an invention or find new applications for the invention.[118] Multiple parties will tend to enter races to develop follow-on improvements or new applications of an innovation if the innovation signals the possibility of other related advances. Monopoly rent dissipation losses will be minimized if a patent

---

[116] *See id* at 316–17 (stating one theorist's view that patent laws may have the positive effect of preventing or reducing duplicative improvement efforts thereby avoiding rent dissipation).

[117] *See United States Patent and Trademark Office, supra* note 6, at 30 (statement of Lee Hollaar, Professor of Computer Science, University of Utah) (describing the duplication of computer programming research resulting from the Patent Office's formerly restrictive views of software patents and the corresponding failure of the patent system to build up a body of published patents describing the developing engineering art in this field).

[118] *See* Grady & Alexander, *supra* note 63, at 308 (pointing out a type of rent dissipation that takes place following the creation of an invention).

holder controlling the invention can bar duplicate improvement and application searching; thereby ensuring that searches that do occur produce the greatest functional gains with the least possible expense.[119] Under this view, patent protections are particularly important for inventions that signal a large possibility of subsequent improvement or application expansion and therefore raise a large risk of duplicative post-invention improvement or application expansion efforts.

Monopoly rent dissipation may also occur where innovators invest too much in keeping their inventions secret.[120] The public can suffer from expensive secrecy measures either because the full benefits of an invention are not made available to the public due to secrecy limitations or because secrecy measures add to the cost of a product or service. Patent protections can eliminate these costs by creating protections against misappropriation of a patented invention that substitute for physical secrecy arrangements.

Software patent protections may prevent several types of monopoly rent dissipation. Following a software developer's patenting of a particular software advance, other individuals working on efforts to develop similar software advances (or working on different software that they recognize will be far less effective than the patented software) will be deterred from further, wasteful efforts. As even one duplicative software development and coding project may entail significant expenditures, the prevention of even a few wasteful coding efforts may achieve substantial societal gains.[121]

The potential benefits of patents in deterring duplicative software development efforts are undercut somewhat by the delays which may occur in the issuance of software patents. When a software developer produces a new type of innovative software and seeks a patent on this advance, the developer's patent application generally will not be disclosed publicly at the time the application is filed. Rather, the application and the advance that it reflects will remain secret until a patent issues or the contents of the application are otherwise disclosed. This may be several years after the patent application is filed. In the

---

[119] *See id.* at 315.

[120] *See id.* at 318.

[121] *See id.* at 319–20 (arguing that the case for patent protection is strong where the protection is likely to discourage costly forms of duplicate innovation).

period while the patent application is pending and secret, duplicative software development by parties other than the patent applicant may continue. This may be mitigated somewhat if a patent applicant is sufficiently confident of her ultimate receipt of a patent to disclose her invention and her related patent application as of the filing of the application, thereby signaling to other developers that further efforts to develop similar software will probably be restricted by the first innovator's patent.

Software patents may also prevent duplicative efforts and monopoly rent dissipation in connection with software application search and improvement efforts. Subsequent to the issuance of a patent on a new software advance, software developers other than those working in concert with the patent holder will be discouraged from undertaking duplicative efforts to implement the new software advance in new products and to develop improvements to the advance. These duplicative efforts will be discouraged because the results of such efforts will be restricted and unusable without the patent holder's consent.[122] By designating and working closely with a chosen set of application developers and invention improvement developers (or by undertaking application development and invention improvement efforts itself), a company that holds a patent on a software advance can ensure that reasonable efforts are made to improve and apply the patented advance while preventing—or at least discouraging—duplicative efforts along these lines by other parties.

Since much of the development expense of a software product lies in efforts after the point of conception to produce workable programming code and to improve overall program operation,[123] the prevention of duplicative software improvement or application efforts concerning a particular software advance may avoid significant waste and lost invention value. The avoidance of this type of waste may therefore provide an important rationale for software patent protections.

---

[122] *See id.* at 307 (stating that the original inventor would have rights and benefits to any subsequent developments or improvements unless that right was sold to those persons who improved the invention).

[123] *See generally* Joseph G. Arsenault, *Software Without Source Code: Can Software Produced by a Computer Aided Software Engineering Tool Be Protected?*, 5 ALB. L.J. SCI. & TECH. 131 (1994) (discussing in detail computer programming code program operation, and software development protection).

Whether or not the public gains net benefits from giving a software patentee control over subsequent efforts to apply or improve a patented software advance will depend in part on the predictability of useful directions of product improvement and application expansion. If a patent holder cannot predict the types of research into new improvements and applications of an innovation that are likely to lead to further advances, then patent controls over subsequent software improvements and applications will have negative consequences beyond the prevention of duplicative efforts. These controls will improperly cut off entire types of useful improvements and applications that are not predictable by the patent holder.[124]

Where potential improvements or applications of an initial software innovation are not clearly apparent from the nature of the innovation, the public may be best served by a competitive "free for all" among software innovators to identify new improvements and applications.[125] This competition and the involvement of more developers may produce a broader range of follow on innovations and applications than the original innovator could have foreseen and caused authorized researchers to pursue. In areas where the full scope of improvements or applications of a software innovation are unlikely to be appreciated by the original innovator, concerns over the undesirable curtailment of subsequent improvement and application efforts, rather than countervailing concerns over maximizing monopoly rents by preventing the duplication of such efforts, suggest that software patents should be interpreted to minimize a patent holder's control over subsequent product improvements and applications.[126] This may be achieved by

---

[124] *See* Samuelson, *supra* note 70, at 2330–31 (indicating that innovation in software development is typically incremental in that programmers often adopt ideas from other programs in a new context); *see also* Cohen *supra* note 51, at 17–18 (noting the likelihood that holders of patent rights will not optimally develop potential improvements to patented inventions).

[125] *See* Sumuelson, *supra* note 70, at 2366–67 ("The public benefits from competitive imitation because it results in the production of more goods at lower prices as more efficient producers enter the market."); *see also* Cohen, *supra* note 51, at 17 (recognizing that competing innovators often develop product improvements more actively and efficiently than holders of patent rights who are without rivals).

[126] *See* Samuelson, *supra* note 70, at 2409 (describing why a rational legal regime for protecting software producers should include a means to avoid wasteful reduplications of efforts); *see also* Cohen, *supra* note 51, at 8–9 (arguing that broad interpretations of patent's scope can severely discourage innovation).

interpreting patents narrowly so that they do not cover a broad range of new improvements and applications, or by creating a scheme of compelled, nonexclusive licensing that will ensure that multiple developers are entitled to use patented software in further applications and improvements.

Finally, software patents should produce further societal gains by reducing the need for patent holders to maintain secrecy measures to control access to innovative software. Absent patent protections, a software innovator might implement costly physical security measures limiting access to a software advance or pursue expensive efforts to enforce confidentiality agreements or trade secret rights which obligate others to maintain the secrecy of an advance. These measures may involve both direct costs of administration[127] and further costs in impaired effectiveness of the affected software as it is used imperfectly under physical or confidentiality constraints. These types of costs would be avoided under a system of patent rights that permits software innovators to impose legal limits on the use of software advances without the need to constrain access to those advances with physical or contractual secrecy measures.

III. SPECIAL GROUNDS FOR RESTRICTING SOFTWARE PATENTS

The policy considerations addressed in the preceding section suggest that software patents are likely to realize many of the same public benefits that patents produce in other technological fields. These considerations tend to support broadly inclusive patentability standards for innovative software. There may be countervailing considerations peculiar to software patents, which justify a more restrictive approach.[128] This section will examine

---

[127] *See* Christopher S. Cantzler, *State Street: Leading the Way to Consistency for Patentability of Computer Software*, 71 U. COLO. L. REV. 423, 436 (2000) (commenting on administrative difficulties in using software licensing agreements to protect against eventual copying or re-use of particular parts of programs).

[128] *See* Samuelson, *supra* note 70, at 2367–68 ("[I]nformation products, such as computer software, bear so much of the technical know-how required to make them on or near the surface of the product that natural lead time for this kind of industrial product may not suffice."); *see also* Pamela Samuelson, *Benson Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025, 1113–33 (1990) (describing the need for special patent laws governing software in light of the dual expressive and functional nature of software code).

some special concerns that have been raised about software patents, with an emphasis on how those concerns may be reduced by narrowly targeted limitations on software patentability.

Previously articulated concerns about software patents have focused on three types of problems: 1) the possibility that patents on intangible software innovations, which lack physical features and tangible structural elements, may restrict information processing sequences and related intellectual processes,[129] 2) the chance that software patents may issue for innovations lacking identifiable utility and corresponding societal gains,[130] and 3) the potential impact of patent enforcement in discouraging or limiting desirable software development.[131]

A. *The Absence of Tangible Structural Features*

Some software innovations reflect little more than computer implementations of new intellectual concepts or relationships.[132] These innovations differ from many prior technological advances in that these software advances lack physical features or tangible structural elements. The value of these software advances derives not from their physical characteristics or physical operations, but rather from the increase in information value that is achieved by computer processing controlled through the innovative software.[133]

---

[129] *See infra* notes 132–158 and accompanying text; *see also United States Patent and Trademark Office, supra* note 6, at 24 (statement of Jerry Baker, Senior Vice-President of the Oracle Corp).

> A complex program may contain numerous established concepts and algorithms as well as a multitude of innovative ideas. Whether a software program is a good one does not generally depend as much on the newness of each specific technique, but instead depends on how well these are incorporated into the unique combination of known algorithms and methods. Patents simply should not protect such a technology.

United States Patent and Trademark Office, supra note 6, at 24.

[130] *See infra* notes 159–182 and accompanying text; *see also* U.S. CONST. art. I, § 1 cl. 8 ("To 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.").

[131] *See infra* notes 183–220 and accompanying text; *see also United States Patent and Trademark Office, supra* note 6, at 7 (statement of Douglas Brotz, Adobe Systems) ("The software industry has thrived without patents.").

[132] *See* Samuelson, *supra* note 70, at 2330 (pointing out that innovation in software development is typically incremental).

[133] *See id.*

The capability of general purpose computers to be specially programmed to undertake new information processing steps means that a new information processing idea can often be translated into an equivalent computer-based system for processing information in accordance with the new idea.[134] The

---

[134] While computer implementations of software advances all involve some physical features beyond the programming instructions comprising the software, the nature of these further physical features can vary widely. The information processing steps described in a software program can be implemented through several types of hardware components. One type of implementation involves a general purpose computer that is directed to process information by the software. A second, different type of hardware implementation involves a highly specialized circuit designed to implement in wiring the same sequence of information processing steps that are defined in the software. These two types of design approaches—one relying on a general purpose electronic device with specific instructions for dealing with a given information processing task and the other with a task-specific electronic design—represent the end points of a spectrum of possible designs for implementing a single information processing sequence. One leading intellectual property attorney has given the following description of the continuum of potential information processing devices between these two extremes:

> Computer-implemented solutions to technological problems in the form of processes and/or machines typically exist along a design spectrum, ranging from pure hardware, that is random logic, to pure software, that is an externally-loaded computer program running on a general purpose digital computer.
>
> Intermediate points along the spectrum involve designs which may be described as special purpose computers and which combine elements of hardware and software in varying proportions, using random logic, array logic, such as PLAs and PALs, microcode and firmware, firmware being fixed programs stored in internal read-only memory.
>
> The particular point along the design spectrum that represents the optimum solution to a given problem is determined by a variety of factors, such as cost, speed, size, flexibility and so on. Moreover, the optimum design point moves over time as competing implementation technologies evolve at different rates. For example, in the mid '70s, complex video game functionality was implemented entirely in random logic. After the arrival of the microprocessor, the very same functionality was realized using firmware.
>
> Finally, technologies such as logic synthesis are becoming available, by which a software solution can be "translated" into an equivalent hardware solution, and vice versa. It should be self evident that as a matter of legal policy, [the patentability of a computer-based innovation should not depend on which of these implementation approaches is used since] the law should not promote artificial distinctions that the technology does not recognize.

United States Patent and Trademark Office, *supra* note 6, at 57 (statement of Ronald S. Laurie, Attorney, Weil, Gotshal & Manges); *see also id.* at 63 (statement of Gideon Gimlan, Attorney, Fliesler, Dubb, Meyer & Lovejoy) (noting that, for

resulting system will obtain its utility largely from the new information processing idea which underlies the system and the gain in functionality that is achieved by implementing the new idea.[135]

In these circumstances, where the primary inventive effort and practical utility associated with a new computer program are related primarily to the development of new information processing ideas, the program may not warrant patent protections because it does not reflect the type of technological advance that patents were intended to encourage and reward.[136] A patent may not be appropriate in these circumstances because the software involved does not involve a substantial advance in the design of physical structures and elements that are combined in some new way to achieve a practical result. Instead, this type of software, as it is based on new conceptions of information processing tasks, primarily reflects a breakthrough in underlying abstract knowledge.[137] Such software should arguably be unpatentable because, like the abstract knowledge on which the software is based, the new program makes no new contribution to public knowledge about the construction or operation of useful devices or physical processes.[138] In short, a

---

purposes of assessing patentability, "it should make little difference that an invention is implemented in hardware, software, or in-between-ware. In the eyes of the electronic circuits that carry out a given invention, there really isn't any functional difference").

[135] Functionality is the hallmark of a patentable invention. *See* Diamond v. Diehr, 450 U.S. 175, 185 (1981) ("Excluded from such Patent Protection are laws of nature, natural phenomenon, and abstract ideas."). The functionality of information processing advances should generally determine their patentability, and not the degree of hardware or software used to implement that functionality. Indeed, the functionality of an information processing system may be its only distinctive, invariant feature since the system's hardware and software details will frequently be varied to adjust the system to particular tasks. As described by one experienced patent attorney:

> I often say that electronics is applied functionality in the electronic domain. Implementation of this functionality is merely a design choice in most cases in terms of hardware or software. The choice of how to implement this inventive functionality depends on many factors, but it is the functionality that is the invention, and one should never forget that.

United States Patent and Trademark Office, supra note 6, at 77 (statement of Robert Green Sterne, Attorney, Sterne, Kessler, Goldstein, and Fox).

[136] *See Diehr*, 450 U.S. at 185 (quoting Rubber-Tip Pencil Co. v. Howard, 20 Wall 498, 507 (1874)) ("An idea is not patentable.").

[137] *See* Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980) (indicating that abstract ideas, such as Newton's law of gravity, are not patentable).

[138] *See* Parker v. Flook, 437 U.S. 584 (1978) ("A principle in the abstract, is a

developer of software which is new and innovative only in that the software incorporates a new information processing idea may not have made a patentable contribution to public knowledge about physical technology and engineering. Without such a contribution, the developer may not have satisfied his or her part of the quid pro quo bargain underlying patent rights.

Patent rights are enforced to encourage technological design efforts—that is, studies of the nature of practical problems, the functional characteristics of various devices and processes that can address practical problems, and the specification and public disclosure of new combinations or "structures" of device components or procedural steps which will solve practical problems.[139] The ultimate goal of these incentives is to give the public new, artificially-created tools, that aid individuals and other parties in solving problems and undertaking useful tasks.

---

fundamental truth; an original cause; a motive; these cannot be patented, as no one can claim in either of them an exclusive right.").

[139] While special types of patents may be obtained for new decorative designs of industrial products and new plant varieties, *see* 35 U.S.C. §§ 161, 171 (1994), most patents are "utility patents," which are granted for useful items and processes. Utility patents are only available for new or improved designs of processes, machines, items of manufacture, or compositions of matter. 35 U.S.C. § 101 (1994). New designs for machines, items of manufacture, and compositions of matter clearly involve new specifications of physical details of the invented items. As to new processes, some courts have looked for a physical transformation in either the means or ends of process steps before deeming a process to be patentable subject matter. *See, e.g.*, Cochrane v. Deemer, 94 U.S. 780, 788 (1877) ("[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 or thing."); In re Durden, 763 F.2d 1406, 1410–1411 (Fed. Cir. 1985) ("A process . . . is a manipulation according to an algorithm . . . doing something to or with something according to a schema."). Recent judicial analyses of software innovations, however, have suggested that a physical transformation is not a required feature of patentable subject matter. Patentable subject matter is present, according to these analyses, if an invention produces useful results in a regularly operative, distinctly described way. For example, in State St. Bank & Trust Co. v. Signature Fin. Group Inc., 149 F.3d 1368 (Fed. Cir. 1998), *cert. denied*, 119 S. Ct. 851 (1999), the Court of Appeals for the Federal Circuit summarized its approach to software patentability as follows:

> Today, we hold that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces 'a useful, concrete and tangible result'—a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.

Id. at 1373.

Patent incentives encourage engineers to go beyond intellectual ideas such as scientific or mathematical principles to create artificial tools that operate in accordance with those principles. The scientific and mathematical principles embodied in a new invention—even if the principles themselves are newly discovered—are not protected by a patent on the invention. Rather, the patent applies to and rewards the specification of the additional elements of the invention which allow the abstract ideas or knowledge about a physical phenomena or context to be applied to produce practical results.

For example, Newton's famous theorem stating that F=ma (force equals mass times acceleration) would not be patentable even if discovered today. A new design for a scale that specified a set of physical elements for measuring the downward force achieved by an object reacting to the pull of gravity and that relied upon the relationship stated in Newton's theorem to translate this type of force measurement into a measurement of the mass of an object would be patentable if the scale design was novel and a non-obvious extension of prior scale designs. The specification of the physical means to gather the information needed to apply Newton's theorem and the specification of the further physical means for recording or displaying the mass size results are the sorts of engineering efforts beyond scientific understanding that patent incentives and rewards were intended to encourage. Without the addition of these physical elements, society has only an abstract theory relating force (F) conceptually and descriptively to mass (m) and acceleration (a). With the addition of these further elements, society has a new and useful tool and the technological arts are advanced. In short, engineering discoveries constituting patentable subject matter are present where abstract ideas are related to the physical world through technological design efforts which describe how physical actions or materials will achieve useful results or how intangible actions can measure or manipulate external physical contexts in useful ways.[140]

A software advance that involves no more than a new sequence of information processing steps may lack patentable

---

[140] *See* 35 U.S.C. § 101 (1994) ("[W]hoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.").

subject matter because the advance is merely a new mode of analysis without a specification of how that type of analysis relates to a practical or useful result. Software patents for such advances may be unwise for two reasons. First, patents on such incomplete designs may reward innovators for creating and disclosing basic analytic constructs—the equivalent of new scientific theories or mathematical algorithms—rather than practically useful tools. Second, patents on intangible software advances may be interpreted broadly so as to impose undesirable restrictions on equivalent mental processes.[141]

The potentially adverse impact of software patents on intellectual activities and mental processes raises questions about these patents that do not arise for patents in other technological areas:

> We all know that software is somehow different from all traditional inventions. But how does it differ from the devices that are surely what the framers of the Constitution envisioned when they mandated patent protection? The difference is that all traditional inventions enhance our physical capabilities, whereas software mimics the mind and enhances our intellectual capabilities. This is what makes software different from all patentable devices and this is what justifies *sui generis* treatment of software under intellectual property laws.
>
> Let me define what software is for the purpose of our discussion, based on its functionality, its utility and the useful character of its art: Software is what occurs between stimulus and response, with no physical incarnation other than as representations of binary logic.
>
> The fundamental question is: Do we want to permit the monopoly possession of everything that works like logical intellectual processes? I hope not.
>
> The mind has always been sacrosanct. The claim that intellectual processes of logical procedures that do not primarily manipulate devices, as in *Diamond v. Diehr*, can be

---

[141] *See In re* Warmerdam, 33 F.3d 1354, 1360 (Fed. Cir. 1994) (rejecting the patentability of an idea which is "nothing more than the manipulation of basic mathematical constructs, the paradigmatic 'abstract idea' "); *see also* Steven M. Santisi, *In re Qarmerdam: When is a Software Process Too Abstract to Merit Patent Protection?* 13 J. MARSHALL J. COMPUTER & INFO. L. 667 (2000) (contending that by equating "abstract ideas" with "the manipulation of abstract ideas," the court "improperly focused on 'how' the process works rather than on 'what' the process does").

> possessed and monopolized, simply extends greed and avarice much too far.
>
> What frightens and infuriates so many of us about software patents is that they seek to monopolize our intellectual processes when their representation and performance is aided by machine.[142]

Concerns about the possible impact of patents on mental and intellectual processes are not new. For some years, courts have applied the "mental steps doctrine" to prevent the patenting of certain information processing advances lacking physical features.[143] This doctrine provides that processes are unpatentable if the processes involve only mental activities or equivalent information processing steps.[144] The mental steps doctrine has been invoked primarily to deny patents to inventions consisting of new mathematical formulae or methods of computation.[145]

At one point, patents on software-based advances were evaluated under the mental steps doctrine by determining if the innovations described in the patent claims included physical details or elements that distinguished the claimed inventions from pure mental processes.[146] As applied to software, the

---

[142] *United States Patent and Trademark Office, supra* note 6, at 48 (statement of Jim Warren, Autodesk, Inc.).

[143] *See, e.g.*, Diamond v. Diehr, 450 U.S. 175 (1981) (explaining that having one or more steps that are not novel is irrelevant to the issue of patent protection of the whole); *In re* Abrams, 188 F.2d 165, 169 (C.C.P.A. 1951) ("[P]urely mental acts are not proper subject matter for protection under the patent statutes. . . ."); *In re* Shao Wen Yuan, 188 F.2d 377, 380 (C.C.P.A. 1951) (asserting that it is thoroughly established that mental steps do not form a patentable process. *See generally* DONALD S. CHISUM, PATENTS § 1.039(6) (1996).

[144] *See, e.g.*, *Diehr*, 450 U.S. at 185 ("Excluded from such patent protection are laws of nature, natural phenomena, and abstract ideas."); *In re* Heritage, 150 F.2d 554, 556 (C.C.P.A. 1945) ("[P]urely mental acts are not proper subject matter for protection under the patent statutes . . . ."); *In re* Shao Wen Yuan, 188 F.2d at 380 ("[P]urely mental steps do not form a process which falls within the scope of patentability as defined by statute").

[145] *See, e.g.*, Musco Corp. v. Qualite, Inc., 1997 U.S. App. LEXIS 790, at *6–8 (Fed. Cir. 1997) (discussing mental steps process); Lyman v. Ladd, 347 F.2d 482, 483 (D.C. Cir. 1965); Don Lee, Inc. v. Walker, 61 F.2d 58, 66–67 (9th Cir. 1932); *In re* Bolongaro, 62 F.2d 1059, 1060 (C.C.P.A. 1933); *In re* Shao Wen Yuan, 188 F.2d at 379–380.

[146] *See* Robert Hulse, *Patentability of Computer Software After* State Street Bank & Trust Co. v. Signature Financial Group, Inc.*: Evisceration of the Subject Matter Requirement*, 33 U.C. DAVIS L. REV. 491, 504 n.86 (2000) (discussing the Freeman-Walter-Abele test, where courts have held that a "[c]laim is patentable only if it applies [an] algorithm to physical elements or steps of a process").

mental steps doctrine was used to restrict software patent rights and incentives to innovations with at least some physical structures that were critical to the operation of the innovations.[147]

The types of physical elements or interactions with a physical environment that are sufficient to overcome the restrictions of the mental steps doctrine were clarified in *In re Prater*.[148] In that case, the Court of Customs and Patent Appeals interpreted the mental steps doctrine narrowly to preclude only patents that would directly interfere with the free use of mental processes.[149] The court concluded that the mental steps doctrine should only restrict the patentability of information processing sequences standing alone—that is, the mental steps doctrine should apply only if patent protection is sought for an information processing sequence per se, such that a mental process incorporating that sequence would fall within the patent and apparently be restricted.[150]

Patent claims that are expressly limited to computer-related or computer-implemented versions of an information processing sequence do not run afoul of this narrow version of the mental steps doctrine. These computer-based processes constitute patentable subject matter because they do not involve mere mental steps, but rather combinations of information processing steps and computer activities. Mental steps, even those accomplishing equivalent information processing, will be restricted by this type of patent since only persons implementing the specified information processing with a computer will infringe these patents. In short, recognizing valid patents on computer-based information processing systems and procedures creates patent rights and restrictions only in computer-based versions of the information processes

---

147 *See* PETER D. ROSENBERG, PATENT LAW FUNDAMENTALS § 6-01 (2d ed. 2001) (noting that the mental steps doctrine was applied to preclude the patenting of mental acts or steps, "which may be performed by the human mind without the need or intervention of physical instrumentality"; examples of unpatentable mental steps include processes for computing, measuring and determining information); *see also* Arrhythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053, 1059 (Fed. Cir. 1992) (stating that transforming one electrical signal into another is a physical process).

148 415 F.2d 1378 (C.C.P.A. 1968), *modified on reh'g*, 415 F.2d 1393 (C.C.P.A. 1969).

149 *See In re Prater*, 415 F.2d at 1389.

150 *See id.*

involved, leaving persons free to use purely mental versions of the same processes.

This narrow interpretation of the mental steps doctrine[151] ensures that patent incentives remain in place for two types of information processing advances that are implemented through new software. First, innovation in applying old information processing ideas in new software contexts or uses is encouraged. Even where a particular mode of information processing or mental analysis is already known, patent protections and incentives will reward software developers for producing new, non-obvious implementations in innovative software of these older information processing ideas and mental steps.

Second, innovation in identifying and applying new information processing ideas that are suitable for software implementation is also encouraged. Patent incentives for the creation of new, non-obvious software will indirectly encourage the development of new information processing ideas which can form the basis for such software. Of course, a patent covering software that is based on a new information processing idea may restrict the use of the same idea in further software. Such a patent will not, however, limit the use of the same idea in mental processes or in engineering designs not involving software.[152] Through public disclosures of the new information processing idea in a patent on related software, society gains two types of benefits: access to the patented software (albeit under patent restrictions during the term of the patent) and free use of mental applications of the new idea in other contexts.

The difficult remaining question is what sorts of physical or

---

[151] *See In re Prater*, 415 F.2d at 1393; *see also* Gottschalk v. Benson, 409 U.S. 63, 63 (1972) (holding that claims are not patentable if they are merely a series of mental steps); *In re* Meyer, 688 F.2d 789, 789 (C.C.P.A. 1982) (holding claims not patentable as they represented a mental process); *In re* Chatfield, 545 F.2d 152, 152 (C.C.P.A. 1976) (holding the claims patentable, as they were not drawn to purely mental steps); *In re* Bernhart, 417 F.2d 1395, 1395 (C.C.P.A. 1969) (holding claims patentable, because no "mental steps" issue was involved and the claims did not encompass mere mental activity).

[152] The patent may limit the use of the same idea in electronic circuit designs that may implement information processing that is equivalent to the processing performed by a computer programmed with the patented software. The particular electronic means used to achieve the information processing will not be material. All of these means of implementing the same information processing will probably be seen as equivalents of the software-controlled computer processing and will be found to infringe a patent on the software.

other features in a software-based advance should distinguish a patentable application from an unpatentable restatement in programming code of information processing sequences that can also be implemented in mental steps. The latter remains unpatentable under even the narrowest interpretation of the mental steps doctrine.[153]

Sequences of information processing can be part of both patentable software applications and unpatentable software implementations of mental step equivalents. Consequently, the presence of substantial information processing features in a software design generally will not provide a basis for determining if an unpatentable mental step equivalent is present.

There is, however, one important exception to this rule. Information processing steps that are particularly unsuitable for human mental activity should be deemed proper bases for patentable software regardless of whether a practical application of the steps beyond the operation of a computer is shown. Where information processing steps in a software product are so extensive or quickly performed that they are impossible or highly unlikely to be undertaken through human mental activities, then these information processing steps should be considered to be solely suited to computer processing. A software product that includes such computer-specific information processing steps should be viewed as patentable subject matter without limitation under the mental steps doctrine. Such a program implements information processing techniques that go beyond mere mental steps. The enforcement of a patent on a software product incorporating these types of computer-specific information processing procedures will have no negative impact on mental activities since the information processing steps that will be limited by the patent are ones that would probably not be used in mental activities anyway.

If, however, the information processing steps embedded in a new software product are ones that could realistically be

---

[153] *See In re* Abrams, 188 F.2d 165, 169 (C.C.P.A. 1951) (holding that purely mental steps are not patentable). *But cf. In re* Musgrave, 431 F.2d 882, 890 (C.C.P.A. 1970) (stating that if "pure mental steps" is construed as encompassing only steps incapable of being performed by a machine, then deeming them not patentable is the correct result, but if "pure mental steps" is construed as to encompass steps performable by machine, as well as mentally, then deeming the claims not patentable is unsound).

undertaken by both humans and computers, the software product will need to include other functionally significant elements in order to be patentable subject matter. These additional functional elements will serve two important purposes related to assessments of patentable subject matter in software innovations.

First, the functional elements that are added to the software's information processing sequences will distinguish patentable software from mere restatements in programming code of information processing steps. Patent claims will need to draw on this distinction by including in descriptions of software sought to be patented specifications of both the software's information processing steps and the functional features linking the software to the solution of a practical problem. A patent containing claims to a software invention described in this way will only be infringed if a product or process includes a similar combination of information processing and functional elements. Patents drafted in this manner will prevent others from replicating or reusing the protected software in a functionally similar manner, but will not prevent others from using the same information processing steps in mental processes.[154] The latter will remain unaffected and freely available because these mental steps will not involve the functional features of the patented invention. Because mental steps remain unaffected, this approach to protecting software innovations satisfies the concerns underlying the mental steps doctrine.[155]

Second, the functional elements of a software-based innovation provide evidence that the innovation is a practical, technological advance rather than just an improvement in methods for abstract information handling or analysis. The functional elements of a software-based innovation will typically relate the information processing undertaken through the innovation to physical results or other practical consequences. These functional elements often account for the practical value of

---

[154] *See* Diamond v. Diehr, 450 U.S. 175, 191–193 (1981) (explaining that a practical process including a mathematical equation is patentable, because even if a patent holder restricts the use of the process, others are free to use the equation).

[155] *See* Jur Strobos, *Stalking the Elusive Patentable Software: Are There Still Diehr or Was it Just a Flook?*, 6 HARV. J.L. & TECH.. 363, 369–370 (1993) (describing a paradigm where computers add value in automating previously used calculations, the added value is the subject of the patent and there is no preemption of human processes).

an innovation. As the source of this practical value, these functional elements are critical features of patentable subject matter in software-based innovations.

Several types of functional elements should be sufficient to create patentable subject matter when added to sequences of software-controlled information processing. One type of functional element that should generally be sufficient is a functionally significant interaction between software-controlled information processing steps and physical features of the computer or computers carrying out those steps. To protect this type of combination, patent claims will need to describe a claimed invention in terms of a series of information processing steps which manipulate computer operations in a specified manner and the specified sequence of computer operations must have practical utility. Examples of sufficient utility include increases in the speed or accuracy of computer processing. The enforcement of this type of patent will have no impact on the use of similar information processing in mental steps or other activities since these other uses will not involve the required manipulation of computer processing and will not infringe on this type of patent.

Software-controlled information processing can also be combined with physical elements of an innovation in a second way that will create patentable subject matter and distinguish the innovation from a mere information processing sequence. If a software-based advance is used as a means to measure or control a physical item or phenomena, a patent can be sought for the combination of the software-dictated information processing steps and the functional elements that obtain measurement inputs about physical surroundings or apply control outputs to physical tasks. This sort of combination of software processing and related computer input or output elements will be distinguishable from mental steps involving the same patterns of information processing since these mental steps will lack the physical input or output features of the patented invention. Furthermore, these physical input or output elements will provide a means to apply software–controlled information processing to a practical task, thereby achieving the practical utility which is necessary for a patentable advance.[156]

---

[156] *See id.* (describing a paradigm that recognizes that although computers

Of course, information processing patterns used in a computer program and the means used to gain inputs for the program or to apply the program's outputs to a practical task may sometimes be used by humans through manual processing that does not involve a computer. For example, a new method for measuring the intensity of light might initially be developed with the use of a computer to interpret electronic measurements of light, but later be performed manually by individuals who complete the calculations and information processing formerly handled by the computer. A patent on the initial computer-implemented version of such an information-handling process for measuring a physical phenomena might also be deemed to cover and limit the human-performed version of the same process. That is, the human-performed version might be seen as infringing the patent on the computer-implemented version of the process because the human-conducted version is equivalent in every key respect to its computer-based predecessor.[157]

In these circumstances, a patent originating from a computer-based process might limit mental processes used to complete the same process. This narrow impact on practically applied mental processes only arises because the mental processes involved are used in the context of a practical task and application controlled by the patent. The limitation of applied mental processes in this way is no different than would occur were the human-performed version of the process patented at the outset. Since the human-performed version of the process is more than a mere mental process in that it has further practical elements applied to accomplish a useful task, a patent on this human-performed process would not be barred by the mental steps doctrine. The same should be true regarding patents on

---

duplicate existing inventions, they add value in the way of speed, volume, accuracy, convenience and automaticity, and this added value is the subject of the patent); *see also* Diamond v. Diehr, 450 U.S. 175, 191–193 (1981) (finding calculations that could not be patented by themselves were nonetheless patentable when they were used to control the physical features of a rubber molding process).

[157] *See, e.g.*, Scott Lund, *Patent Infringement and the Role of Judge and Jury in Light of Markman and Hilton Davis*, 21 IOWA J. CORP. L. 627, 633 (1996) (stating that although a patent is not literally infringed, courts will find a patent infringement where two patents are nearly equivalent); Allen Newell, *Symposium: The Future of Software Protection: Response: The Models are Broken, The Models are Broken!*, 47 U. PITT. L. REV. 1023, 1026–28 (1986) (expressing concern that broad patent protection for software-implemented algorithms could prevent individuals from undertaking similar information analyses).

computer-based processes even where the patents may have some later impact in limiting human-conducted processes and related mental steps.

In sum, the above analysis suggests that there are at least three types of software-based inventions that have features that clearly distinguish them from mental processes; thus, should be patentable without limitation under the mental steps doctrine. These patentable software advances include the following:

(1) *Inventions based on information processing steps that are not suitable for human mental activity.* An example of this type of invention is information analyses undertaken at computer processing speeds to determine where in a sequence of computer operations a computer presently stands and to predict what next step will be the most efficient for the computer to take. The need for processing speeds far higher than humans are capable of makes it highly unlikely that the information processing sequence involved here would be employed in mental activities. A patent on these computer-implemented processing sequences is unlikely to have any restrictive effect on mental processes.

(2) *Inventions involving information processing achieved through specified changes in computer-processing hardware.* An example of this type of invention is a computer operating methodology in which the sequence of information processing involved was specified not in terms of information transformations but rather in terms of corresponding state changes in individual computer components. Defined this way, only another implementation of the same information processing through similar state changes could infringe a patent on this invention. Mental activities outside of this application context would therefore be unaffected.

(3) *Inventions involving information processing to control or measure another item:* An example of this type of invention is a new scheme of information processing for use in analyzing temperatures in a rubber mold to determine the optimal time for opening the mold. Here, mental activities using the same information processing steps for other purposes will fall outside a patent on this invention. Mental steps will therefore be unaffected by such a patent.[158]

---

[158] Mental steps used to analyze a rubber molding process would be restricted by this patent if the steps were used to produce mold opening instructions. This type of incidental burden on mental processes, however, is always present when a patent

B. *The Lack of Physical Results*

A further possible ground for limiting software patents may be that many software advances do not, of themselves, produce physical results and, arguably, lack the sort of immediate physical utility that the patent laws were designed to further. The lack of physical results from a computer system may raise questions about whether the system is sufficiently applied to practical tasks to qualify for a patent.[159] Absent a clear link to an external physical result, a software-controlled computer system may be little more than an active information processing counterpart to an abstract idea or scientific relationship. The system simply records the idea or relationship in the dynamic form of a pattern of information processing. The system will reorganize or analyze information in accordance with the idea or relationship reflected in the system's programming, but the output of the system, of itself, will have no more practical implications than the idea or relationship on which it is based. Where a device or process design is not tied to a specific physical result, it may be desirable to treat the new design as an unapplied and unpatentable recording or demonstration of the information processing ideas or relationships on which the design is based.[160]

The law withholds patents for discoveries of potentially useful abstract ideas or scientific relationships in part because such discoveries still need to be translated into practical applications. It is desirable to withhold patent rewards in cases where an innovator has discovered no more than a new design

---

covers a physical process. For example, a patent on a particular method for manufacturing a fender will tend to discourage and limit the mental steps associated with the manufacturing method if those mental activities are desired to be undertaken by parties not licensed to complete the patented process. The law permits this incidental burden on mental steps in order to grant patent controls and rewards for the related manufacturing process. A small limitation of this sort on narrow forms of mental activity is probably a necessary price to pay for patent rights and incentives concerning human-controlled processes and practices.

[159] *See* Zoe Milak, *The Copyrightability of Encryption Methods and Encryption Algorithms on Computers*, 1996 U. CHI. LEGAL F. 589, 598–99 (1996) ("[Section] 101 of the Patent Act describes patentable statutory subject matter .... The statute contains two categories of inventions: process and products .... These two categories mandate a physical action and a physical result.").

[160] *See* Strobos, *supra* note 155, at 383 ("[T]he determination of patentable subject matter in software depends on whether the claim is limited in scope to a novel and unique aspect of processing other than that which can be found in the human mind or in the laws describing other natural processes.").

idea, until sufficient additional design steps are undertaken to produce an invention with practical utility to society.[161] To ensure that patents are only granted for software advances having clear and immediate utility, it may be desirable to withhold patents for software advances until an inventor specifies how these advances can cause a computer to control a physical result or otherwise modify or interpret a physical context.

Awarding a patent for a computer system that undertakes a new mode of information processing, but achieves no useful result as a consequence, would effectively reward an inventor for an incomplete invention that does not deliver to the public the type of technological design information which warrants a patent. This type of unapplied information processing advance is incomplete in two senses. First, the invention lacks design elements that would need to be added to the new mode of information processing to achieve a practical result. The development of these further design features may still require substantial inventive efforts. Patent rewards should be withheld to encourage the completion of these efforts. Second, public disclosure in a patent of a software design that produces no more than an information processing result adds no immediately useful devices or processes to public knowledge. Since the public does not gain any increased utility and incremental benefit from the disclosure of this incomplete design, the costs and inefficiencies of patent restrictions on the software design should not be tolerated.[162]

In *Gottschalk v. Benson*,[163] the Supreme Court voiced its concerns about granting patent rewards for the discovery and disclosure of mathematical algorithms that are implemented in

---

[161] *See* Landscape Forms, Inc. v. Columbia Cascade Co., 113 F.3d 373 (2d Cir. 1997) (holding that a furniture manufacturer's line of furniture could not be protected with a patent because it was nothing more than an idea).

[162] A design may be considered complete if its disclosed features imply but do not state practical applications. It is sufficient that the invention details disclosed in a patent would suggest practical applications to the average practitioner in the field of the invention. *See* Brenner v. Manson, 383 U.S. 519, 531–32 (1966) (holding that a patentable invention must include the discovery of the practical utility of a new item, or its utility must be obvious to persons of ordinary skill in the field). Put another way, a design disclosure should be deemed to disclose and add to public knowledge all the practical applications that it communicates both explicitly and implicitly.

[163] 409 U.S. 63 (1972).

computer software, but not applied to particular useful tasks. The Court unanimously rejected the patentability of a computer program that implemented a particular information processing algorithm on a computer.[164] The Court's analysis suggested to some observers that the Court considered many, if not most, software innovations to be unpatentable in and of themselves.[165] The Court's approach implied that software programs should be viewed as new mathematical or information processing algorithms restated in computer commands and the resulting computer systems, like the algorithms themselves, should be seen as lacking the practical utility necessary for patenting.[166]

The difficulty with the Court's analysis in *Benson* was it failed to distinguish between patentable technological designs, where information is processed in accordance with mathematical or information processing algorithms to produce useful results, and unapplied descriptions of information relationships and information processing algorithms that can be used to construct useful applications. Just because portions of software operations can be described in terms of mathematical algorithms does not mean the software involved cannot achieve a practical, technological result. This suggesting that the inclusion of information processing based on a mathematical relationship or algorithm rendered computer software designs unpatentable created great confusion among lower courts and practitioners. As one leading observer noted:

> The result of this formulation has been over two decades of confusion and inconsistency in the case law involving the patentability of software-implemented processes. The fact is that mathematics is a language, albeit a very precise one, and like other languages can be used to describe concepts and relationships that are technologically applied as well as those of a more abstract nature that are not so applied.[167]

---

[164] *See id.* at 67 (noting the invention involved a method to convert numbers from one internal computer to another).

[165] *See* Allen Newell, *supra* note 157, at 1026–28 (1986) (hypothesizing that allowing patents on algorithms could prevent people from doing calculations); Samuelson, *supra* note 128, at 1059 (noting that the Court's analysis in *Benson* called into question the patentability of computer programs).

[166] *See, e.g.*, Cathy E. Cretsinger, *Annual Review of Law and Technology, I. Intellectual Property;* B. Patent AT&T Corp. v. Excel, 15 BERKELEY TECH. L.J. 165 (2000) ("A claim not limited by any physical element or structure could become, in effect, a patent on an abstract idea or a process thought.").

[167] *United States Patent and Trademark Office, supra* note 6, at 57.

The Court's essential objection to the patent at issue in *Benson* may have been that the patent described an invention which was incomplete.[168] Absent a clear indication of how, in a particular physical context, the new information processing steps described in the patent were to be applied, the patent application in *Benson* arguably lacked a complete description of a key feature of the claimed invention. The patent involved in that case might therefore have been ruled invalid on the ground that the physical features of the invention and its utility in a practical context were ambiguously or incompletely described.[169]

The Supreme Court again considered the problem of the necessary physical features of a software-based invention in *Parker v. Flook*.[170] In *Flook*, the Court concluded that a computer system for monitoring temperature data, analyzing the data, and triggering an alarm signal under certain conditions was unpatentable.[171] One basis for this result was the Court's conclusion that the sole physical steps involved in the invention were mere data recording and analysis steps, making the claimed invention largely indistinguishable from a new computational method for use in a narrow temperature measurement context, coupled with a means to record results of the computation.[172] Since a pure computational method would be unpatentable as an abstract idea or discovery, the mere addition of a physical device for recording the results of the calculation did not produce a patentable invention.[173] Therefore, the Court held it to be unpatentable, in part because the invention carried out calculations and had no further utility than to solve a specialized mathematical problem.[174]

Again, the Court seems to have overlooked the distinction between an unapplied mathematical algorithm describing a sequence of data processing and an application of such an

---

168 *See Benson*, 409 U.S. at 64.

169 *See United States Patent and Trademark Office*, *supra* note 6, at 57 (noting that the real issue raised by the patent in *Benson* was "probably not one of subject matter under Section 101 [of the Patent Act], but rather one of indefinite claiming of the invention under Section 112").

170 437 U.S. 584 (1978).

171 *See id.* at 594–95.

172 *See id.* at 595.

173 *See id.* at 590 (stating that a computational method leading to obvious and conventional post-solution activity is not patentable per se).

174 *See id.* at 594–95.

algorithm in a narrow domain such as operating a fire alarm. The particular invention at issue in *Flook* involved two types of physical relationships to practical activities which distinguished the invention from an unapplied, unpatentable algorithm. First, the invention interpreted a specific physical environment, producing useful signals when that environment reached a condition which indicated a fire. Second, the invention controlled an external result—the activation of a fire alarm—which had practical significance since it indicated the probable presence of a fire. Either of these physical features standing alone should have constituted a sufficient physical element to cause the software-based invention in *Flook* to be viewed as a practical design involving patentable subject matter.

In contrast to its rejection of software patents in *Benson* and *Flook*, the Supreme Court upheld the patentability of a software-controlled invention in *Diamond v. Diehr*.[175] The invention at issue in *Diehr* involved computer calculations based on temperature readings somewhat like the fire monitoring system in *Flook*. The calculations considered in *Diehr*, however, were used to control a more extensive physical result. The process at issue in *Diehr* involved the use of temperature measurements and related computer calculations to determine the proper time for completing a rubber molding procedure and initiating the opening of a rubber mold. This process was held to be patentable subject matter largely because the computational portion of the invention was tied to a clear physical result and context.[176] Indeed, in *Diehr* the invention at issue involved two physical results: the physical act of opening of the rubber mold and the physical object produced by the mold. Each of these physical outputs of the claimed invention—one a physical means of operation and the other a physical result—was probably sufficient to establish that the software-based molding process involved patentable subject matter.

While the *Diehr* case involved the manipulation of a physical result through software, it is unclear whether such a physically transformative result is necessary in order for a software-based invention to be patentable subject matter. Some courts have required this sort of physical effect or impact as a threshold test

---

[175] 450 U.S. 175 (1981).

[176] *See id.* at 184.

for patentability.[177] The Supreme Court implied in *Cochrane v. Deemer* that a patentable process must cause a physical transformation in the materials to which the process is applied.[178] In *Cochrane*, the Court used this test as a means to distinguish technological processes that are proper subjects of patent protection from other socially valuable activities that lack the type of physical utility or result that the patent laws are designed to further. Unfortunately, in 1877 when *Cochrane* was decided, the Court did not recognize that a wide variety of transformations in information and other intangible items might be employed to achieve predictable, useful results and, therefore, regularly useful information-handling processes of this sort might be proper targets for patent incentives in addition to physical advances having similar utility.

Recent court decisions have rejected the view that a physical transformation is required as part of patentable subject matter.[179] Instead, patentable subject matter has been found if a device or process either transforms a physical object to produce a physical result or transforms data reflecting a physical object to produce physically significant measurements or analyses.[180] In the latter type of information processing invention where no physical result is achieved, the required information transformation must change the data into a different state or form that reflects increased utility over the untransformed information.[181] In short, if a software advance solves a practical problem through either the physical entities the software controls through computer operations or through the valuable analyses of physical conditions the software produces when

---

[177] *See In re* Schrader, 22 F.3d 290, 294 n.9 (Fed. Cir. 1994) (stating that the "dispositive issue is whether the claim . . . recites sufficient physical activity").

[178] 94 U.S. 780, 787–88 (1877).

[179] *See In re* Prater, 415 F.2d 1393, 1403 (C.C.P.A. 1969) (stating that the *Cochrane* dicta has been misconstrued as "requiring that all processes . . . must operate physically upon substances.").

[180] *See In re* Abele, 684 F.2d 902, 908–09 (C.C.P.A. 1982) (holding that an algorithm applied to attenuation data that represents CAT scan images of physical objects is patentable); *In re* Taner, 681 F.2d 787, 790 (C.C.P.A. 1982) (holding that mathematical applications to seismic data are properly patentable).

[181] *See, e.g.*, AT&T Corp. v. Excel Communications, Inc., 172 F.3d 1352, 1360 (Fed. Cir. 1999) (indicating how patentability here turns on "whether the algorithm is applied in a practical manner to produce a useful result"); Arrythmia Research Tech., Inc. v. Corazonix Corp., 958 F.2d 1053, 1056 (Fed. Cir. 1992) (stating that an essential criterion for the patentability of inventions involving mathematical algorithms is whether such invention "is directed to a new and useful process").

combined with computer operations, the advance should be treated as patentable subject matter.[182] This sort of software advance should qualify for a patent if the software is both a novel and non-obvious improvement over previous software designs and the software is the subject of a timely patent application.

These judicial standards appear to shift the focus of patentable subject matter tests from physical results to practical results. The latter can be either physical or informational. Not all information processing results will be sufficiently practical. New, useful information about physical surroundings or phenomena will be sufficiently practical to make a device or process for producing that information patentable subject matter. Information-handling processes that solve mathematical problems and do nothing more remain unpatentable because these involve neither physical components nor practical results.

With this clarification of patentable subject matter standards to include inventions producing useful information, a wide range of software advances developed to accomplish useful business or personal analyses will be considered to be patentable subject matter. Software for controlling computers to produce information processing results which are useful outside of the computers that produce them will clearly have the practical quality which will qualify the software as patentable subject matter despite the lack of any physical manipulation of external items or activities as the result of using the software.

A further category of software should also be deemed to have sufficient practical results to qualify as patentable subject matter despite the lack of identifiable physical results. This type

---

[182] One commentator on software patent issues has articulated a simpler, but essentially identical test for patentable subject matter in software advances:

> [If a patent] is drawn to the solution of a real-world commercial problem, and the claim functional steps or elements as a whole meet the strict legal requirement to be new, nonobvious and useful, then a patent should issue. The function claimed, not the format, is what is important. It shouldn't matter whether new, nonobvious and useful process steps are claimed in the context of a program or a disk or claimed in a hardware or method format, or in the context of a semiconductor chip. Software-related inventions are valuable to the purchaser not for what they communicate, but for the functions they perform. The functions are what are important and what should be assessed for novelty and nonobviousness.

United States Patent and Trademark Office, *supra* note 6, at 8 (statement of Richard LeFaivre, Apple Computer attorney).

of software includes software controlling computer operations in ways that make those operations more efficient or accurate. This type of software will produce its useful results within the computers that run the software. To the extent that the information processing capacity of a computer is itself a useful resource and this type of software is able to increase that capacity, software for managing or controlling computer activities would appear to be patentable subject matter based on its beneficial and useful impacts on computer operations.

C. *Excessive Restrictions on Software Development*

A variety of commentators have argued that, whatever merits patents have had in promoting the development of other technologies, patent rewards and restrictions are inappropriate means to promote software advances because software innovation differs materially from other types of technological innovation.[183] These arguments against software patent protections have focused on three concerns: first, software patents restricting the use of fundamentally important information processing advances may prevent programmers from incorporating those advances in new software products;[184] second, software patents may protect narrow, poorly publicized advances in software designs that will be difficult to detect and avoid in subsequent software development efforts;[185] and third, improperly issued software patents may significantly impede software development and consumer access for extended periods until these flawed patents are invalidated.[186] Each of these concerns is examined in this section.

1. Restrictions on Fundamental Information Processing Techniques

Many programmers have expressed concern that software

---

[183] *See* John Swinson, *Copyright or Patent or Both: An Algorithim Approach to Computer Software Protection*, 5 HARV. J.L. & TECH. 145, 151–53 (1991); Samuelson, *supra* note 128, at 1113–1133.

[184] *See* Swinson, *supra* note 183, at 151–53.

[185] *See* Samuelson, *supra* note 128, at 1136.

[186] *See generally* Envirotech Corp. v. Westech Eng'g, Inc., 904 F.2d 1571, 1574 (Fed. Cir. 1990) (describing one type of invalid software patent which removes "inventions from the public domain that the public reasonably has come to believe are freely available"); Mohasco Indus. v. E.T. Barwick Mills, Inc., 221 F. Supp. 191, 195 (N.D. Ga. 1963).

patents will restrict software implementations of highly effective information processing methods that should be freely available for inclusion in new software.[187] This concern is based on the view that there are some information processing methods that are so advantageous and important as the basis for practical computer applications that both these methods, and the software necessary to implement them, should remain freely available to software developers and users.[188]

In essence, advocates of this view are arguing that broadly significant and reusable software components deserve the same sort of immunity from patentability that currently applies to abstract knowledge.[189] Just as new abstract knowledge is kept unpatentable in part because we wish to ensure that engineers and other analysts are able to use the knowledge as a design tool in creating practical devices and processes, so too may certain software be so broadly useful and important in the construction of further software that patents should be withheld to permit developers to use this software as a component of subsequent designs without any fear of patent restraints.

The desirability of making abstract information processing ideas and related computer processing methods available to all persons was recognized by the Supreme Court in *Gottschalk v.*

---

[187] *See* David A. Burton, *Software Developers want Changes in Patent and Copyright Law*, 2 MICH. TELECOMM. & TECH. L. REV. 87, 87 (1996) (May 18, 1999) <http://www.mttlr.org/voltwo/burton.pdf> (reporting on the results of a poll of computer programmers that found that programmers feel that software patents impede development); Whitmeyer, *supra* note 50, at 1127 ("[P]atents would unduly restrict the incremental, building-block approach common in the software industry . . . .").

[188] *See, e.g.*, *United States Patent and Trademark Office*, *supra* note 1, at 8–9 (statement of Rob Lippincott, Executive Vice President, Interactive Multimedia Association) (describing the potential adverse impact on the development of new multimedia products of software patents restricting the use of new multimedia presentation techniques; such patents may severely constrain both the developers of multimedia products and the users of those products whose expression may be limited by the lack of freely available multimedia tools); S. Carran Daughtrey, *Reverse Engineering of Software for Interoperability and Analysis*, 47 VAND. L. REV. 145, 178 (1994) (explaining that patent and patent-like protection for software discourages public access to information and provides too much protection).

[189] *See* Gottschalk v. Benson, 409 U.S. 63, 67 (1972) (explaining that an abstract principle cannot be patented); Robert M. Hunt, *You Can Patent That? Are Patents on Computer Programs and Business Methods Good for the Economy?*, BUSINESS REVIEW First Quarter 2001, at 14 (concluding that extending patent protection to computer programs and business methods implemented through computers may or may not spur innovation).

Benson.[190] There, the Court unanimously rejected the patentability of a computer program involving no more than a particular mathematical procedure or information processing algorithm implemented on a computer.[191] The Court indicated that the information processing advance at issue in *Benson* was unpatentable because to issue a patent for this invention would be to "preempt" all use of the information processing algorithm involved.[192] What the Court seemed to be saying, however, was not that the underlying algorithm would be preempted from use—the algorithm could still have been used in non-computer analyses even if the disputed patent was enforced—but rather that the patent would improperly restrict the use of a highly useful mode of computer-processing based on the algorithm.

In short, the Court seemed to say that, just as a scientific principle like the law of gravity or a mathematical relationship such as the Pythagorean Theorem is so basic a tool for the design of useful products and processes that it should be freely available for use by all innovators without patent restrictions, a widely applicable computer processing method or sequence should be freely available for use in later device and process designs. In this analysis, the Court may have perceived that computer processing based on a scientific principle or mathematical relationship will tend to be useful in solving practical problems over the same broad range of settings where the scientific principle or mathematical relationship is useful as an analytic construct.

While a patent on broadly applicable new software may indeed cause the patent to be functionally restrictive and economically significant in a variety of programming contexts, this correlation between application scope and patent importance is a desirable rather than abusive feature of software patents. New software designs with numerous applications tend to be ones that most benefit society. It is desirable that patents attach special rewards to the development of broadly applicable information processing methods and related software.

---

190 409 U.S. 63 (1972).

191 *Id.* at 66–67 (stating that the patent sought was for a patent for the process of converting binary coded decimals to the pure binary numerals used in all modern computers).

192 *See id.* at 71–72 (explaining that the information processing algorithm covered by the contested patent had no significant application outside of its connection with computer processing).

Holders of patents on broadly applicable new types of software may be able to realize considerable income from exclusive sales of the patented software or from licenses to produce or use the patented software in various contexts. This is as it should be. Important, broadly useful inventions should produce big rewards. The promise of such rewards will encourage particularly diligent efforts by software innovators to develop and improve software with numerous applications.

Restrictions on product development and use following the patenting of a fundamentally important product feature are not limited to the software field.[193] Patents on breakthrough discoveries such as telephone designs or xerographic processes used in copiers have permitted trailblazing inventors and their companies to exercise a period of dominance and control in their fields or, as an alternative, to obtain substantial financial rewards for allowing other parties to use their inventions.[194] Society tolerates limits on developing and utilizing new technological designs because patent rewards imposing these limits encourage discoveries and disclosures of socially important products and services that would not otherwise exist.[195]

---

[193] Complaints about patent restrictions on important technologies have arisen in diverse technological areas for many years. For example, during the early development of the aircraft industry there were substantial questions raised as to whether the future development of aviation would be seriously impeded if Wilbur and Orville Wright were allowed to enforce patents on their invention. Wilbur Wright defended the enforcement of the Wright brothers' aviation patents as follows:

> When a couple of flying machine inventors fish, metaphorically speaking, in waters where hundreds had previously fished, and spending years of time and thousands of dollars finally succeed in making a catch, there are people who think it a pity that the courts should give orders that the rights of the inventors shall be respected and that those who wish to enjoy the feast shall contribute something to pay the fishers.

MICHAEL A. GLENN, *An Historical Perspective on Patent Protection for Software—Everything Old Is New Again, in* PRACTISING LAW INSTITUTE, COMPUTER SOFTWARE PROTECTION 131, 137 (1997).

[194] As the Supreme Court noted:

> It may be that electricity cannot be used at all for the transmission of speech except in the way Bell has discovered, and that therefore, practically, his patent gives him its exclusive use for that purpose, but that does not make his claim one for the use of electricity distinct from the particular process with which it is connected in his patent. It will, if true, show more clearly the great importance of his discovery, but it will not invalidate his patent.

Dolbear v. American Bell Tel. Co., 126 U.S. 1, 535 (1888).

[195] *See* Russell Moy, *A Case Against Software Patents*, 17 SANTA CLARA

In short, the choice in the software field, as in other technological domains, is not between free and fettered use of new technological designs. Rather it is between having immediate, free access to some new designs that would be developed with or without patent rewards and access to a greater number of new designs that patent incentives are likely to produce, accepting that some of these designs will be temporarily restricted and encumbered by patent rights during the life of the applicable patents. So long as, generally, the new designs that are encumbered with patent rights are those which would not have been developed and disclosed absent the promise of patent rewards, society does not lose access to many, if any, new inventions through patent restrictions. Rather, society simply adds to its pool of useful inventions a few new and often important inventions that are temporarily restricted by patent rights. Those restricted inventions would not have been immediately available to society absent patent rights since these inventions would not have even been developed and disclosed absent the rewards associated with those rights.

Patent law reflects the choice that temporarily encumbered access to a greater number of new technological designs is better than free access to a lesser number.[196] There seems to be no reason to make a different choice about the use of patent incentives to encourage the development and disclosure of new software designs. To decide otherwise based on the short-term advantages of immediate, free access to innovative software is to sacrifice the long-term incentives for software innovation that the promise of patent rights can achieve. As with innovation in other technological fields, society and the patent system should take the long view and accept temporary patent controls over

---

COMPUTER AND HIGH TECH. L.J. 67, 92–93 (2000) (commenting that properly focused patent rights confer a "benefit upon the public that outweighs the evils of [a] temporary monopoly").

[196] One scholar describes the U.S. patent system as follows:

> Thus United States patent law makes a bargain with inventors: it provides nearly absolute protection for a limited term, in exchange for the inventor disclosing the invention completely at the outset of that term and dedicating it to the public when the patent expires. The underlying policy is not to take something pre-existing away from the public, but to encourage inventors to disclose, for eventual public use, things that never before existed.

1 JAY DRATLER, JR., INTELLECTUAL PROPERTY LAW: COMMERCIAL, CREATIVE AND INDUSTRIAL PROPERTY § 2.01 (Law Journal Press 2000).

new and sometimes broadly applicable software in order to realize greater numbers and diversity of such software innovations in the long run.

2. Problems in Detecting and Avoiding Unauthorized Reuse of Patented Software

Some critics of software patents have been concerned that programmers may have problems in identifying and avoiding (or properly licensing) patented software features.[197] Two types of problems related to the documentation and searching of software patents underlie these concerns.[198] The first one concerns the timing of software patent disclosures. A software patent on an advance may issue long after a party other than the patent recipient has incorporated the same advance in new software products.[199] The result may be unexpected patent infringement liability and restricted use of the nonpatent holder's software.

The second problem concerns the difficulty of discovering the full range of patents that may be infringed by a new software product.[200] If a software programmer can not discover the preexisting patent restrictions on design features that are being considered by the programmer for use in a new software product, then those restrictions can not be taken into account in the software design process. If infringement of an unexpected software patent is encountered once software is already in use in

---

[197] *See* Samuelson, *supra* note 128, at 1136 ("No matter how thoroughly or how often one searches the records of issued patents, one can never know when a patent affecting a software product might issue."); *see also* Swinson, *supra* note 183, at 168 ("[P]atents of computer programs are hard to find, and if found, are impossible to understand.").

[198] *See United States Patent and Trademark Office*, *supra* note 1, at 4 (statement of Paul Robinson, Chief Programmer, Tansin A. Darcos & Company) (arguing that the secrecy and delays surrounding software patent applications until they issue as patents or are otherwise published, and the frequent need for programmers to use multiple potentially patented design elements in a single complex software product, create greater problems for programmers in accommodating software patent restrictions than face product designers in other technological fields).

[199] *See* Samuelson, *supra* note 128, at 1136 ("Developers worry that on the day they introduce their newest and most innovative product to the market, a patent might have issued to another firm that would cover some aspect of the just-introduced product.").

[200] "[P]atents can issue on so many subcomponents of computer programs, a future software developer would have to obtain so many licenses from so many different firms for so many different companies . . . ." *Id.* at 1137.

ongoing activities, the results may include: 1) unanticipated financial liability for patent infringement, 2) lost chances to adopt different business activities that do not require the infringing software, 3) lost opportunities "design around" the patented features of the software to produce non-infringing substitutes, and 4) lost chances to negotiate licenses for the patented software free from the threat of patent enforcement litigation.

Software patents that take a considerable period to emerge from the Patent Office are sometimes described as "submarine patents" because they rise up unexpectedly like a submarine from the ocean.[201] A patent application may linger in the Patent Office for a considerable period—typically two to three years—before a patent issues.[202] During this time, the existence and contents of the patent application are not publicly disclosed by the Patent Office, meaning that software developers other than the patent applicant may independently rediscover or copy the patented design without knowledge that a patent application on that design is pending.[203] Once the applicant's patent issues—as a "submarine patent" which emerges unexpectedly out of the Patent Office—other developers and their customers will be required to stop using software that incorporates the patented design or to obtain a license from the patent holder.[204]

---

[201] "A submarine patent is a patent that is granted, or 'emerges,' after a long pendency period in the U.S. Patent Office, during which time others may have unknowingly infringed on the patent." Christopher R. Batztan, *Mandatory Publication of Patent Applications Prior to Issuance of Patents: A Desirable Change in U.S. Policy?*, 18 LOY. L.A. INT'L & COMP. L. REV., 143, 156 (1995).

[202] *See United States Patent and Trademark Office*, *supra* note 6, at 78 (statement of Robert Green Sterne of the law firm of Sterne, Kessler, Goldstein, & Fox). In one study where a sample comprised of 2081 patents was examined in detail, the average patent prosecution period, e.g., from application date to issue date, was found to be 864 days and the median period was found to be 701 days. *See* Mark A. Lemley, *An Empirical Study of the Twenty-Year Patent Term*, 22 AIPLA Q.J. 369, 385 (1994).

[203] One estimate places the typical development time, from formulation to release, for electronic products such as new computer-based products at 6 to 9 months. *See United States Patent and Trademark Office*, *supra* note 1, at 78. Since this is a far shorter period than the pendency of most software patent applications, a party might easily learn of another party's software innovation or independently rediscover that innovation, incorporate the innovation in a product, and initiate marketing or use of the product before becoming aware that the innovation may be restricted by patent rights stemming from a patent application filed by the first software innovator.

[204] One observer has described the "surprise problem" as follows:

The development and adoption of a new product design prior to the discovery of conflicting patent rights may be a particularly important problem in the software field. New software designs having popular features can sometimes be replicated quickly by software developers in products that are produced and distributed in great numbers shortly after the emergence of the first product reflecting the design. Hence, new products based on a software innovation may be widely marketed and begun to be used in large numbers while a patent application concerning the innovation is still pending. If a patent issues covering this design, persons other than the innovator may be forced to stop selling or using products which incorporate the patented design.

One solution to the problem of submarine software patents is to alter the timing of patent application disclosures. This is probably preferable to the broader, more drastic remedy of restricting software patentability. Without undercutting the beneficial development and disclosure incentives raised by the availability of software patents, a rule requiring publication of pending software patent applications relatively soon after their submission to the Patent Office would give software developers more prompt and useful warning about the scope of potential patent rights. With this warning, developers can decide how to proceed so as to avoid conflicts with patent restrictions.

Recently enacted legislation will implement this type of solution not just for software patents, but for all types of patents. This legislation requires the publication of most pending patent applications 18 months after their submission to the Patent Office.[205] Such an early publication requirement will narrow the total period of delay between a software discovery and potential

---

> The industry feels as though it's being kept in the dark for long periods of time, and then surprised by some unanticipated patent jeopardies. This is clearly the result of delayed publication [of patent applications] on the one hand, coupled with the ability in the Patent Office for an applicant to prolong the prosecution for an inordinate period of time .... This creates an environment where surprises become the rule rather than the exception to the industry.

United States Patent and Trademark Office, supra note 6, at 59 (statement of Lee Patch, Deputy General Counsel and Chief Intellectual Property Counsel, Sun Microsystems).

[205] *See* 35 U.S.C. § 122(b)(1) (Supp. V 2000). This legislation will not require the disclosure of patent applications that the applicant certifies will not be the subject of a patent application filed in a foreign country. *See* 35 U.S.C. §122 (b)(2)(i) (Supp. V 2000).

patent rights. By reducing this period of secrecy of a new software invention, the chances of unknowing reuse of that invention before any notice of probable patent rights is given will be reduced as well.

A second and arguably more serious problem may also limit the ability of software developers to create new software products without accidentally incorporating infringing features. Software developers who are creating new software products may find it difficult to identify all of the outstanding patents that apply to planned features of the software under development. The ability of developers to take software patents into account will turn on the successful matching of features in the products under development with the features covered by software patents. If developers can not assess the full range of patents that will be infringed by the developers' planned software designs, the developers may inadvertently create infringing software without a fair chance to realize that they are doing so. In addition to creating unexpected liability for the developers, this type of built in infringement problem may leave users of the resulting software with products or business procedures that they can not use or that they must license from the patent holder under substantial economic duress.

Two types of limitations on patent searches[206] may lead to mistakes in identifying outstanding software patents that are applicable to a proposed software design. First, a developer may find that his or her new software has so many new design elements that the developer can not check all of the new elements against outstanding patents.[207] Without such a check,

---

[206] One commentator has stated as follows:

> While not required to do so, the inventor and his or her lawyer will often have conducted a patent search, either by themselves or through the services on one of many search companies, in order to study the claims of prior patents. In many situations, the search will disclose prior patents which include elements (or similar aspects) of the invention claimed by [the] inventor.

G. GERVAISE DAVIS, III, SOFTWARE PROTECTION: PRACTICAL AND LEGAL STEPS TO PROTECT AND MARKET COMPUTER PROGRAMS 150 (Van Nostrand Reinhold Co. 1985).

[207] This problem was described by an official of the Oracle Corporation as follows:

> The engineering and mechanical inventions for which patent protection were devised are often characterized by large building-block inventions that can revolutionize a given mechanical process. Software seldom includes substantial leaps in technology, but rather consists of adept

the developer will be unable to thoroughly detect and avoid infringing designs. Second, software patents may describe and index protected software features in ways which impair effective searching for relevant software patents when a particular software feature is incorporated in a software product. If software features are not consistently and unambiguously described in software patents, then patent searchers will have a difficult time in comparing the features of a proposed software product with the software features protected by prior patents.

These problems are not, however, unmanageable. The number of outstanding patents that should be checked against a new software product will be limited by the small range of valid patents that are likely to be relevant to the product. As developers create new or modified software designs, only significantly new design elements will generally need to be

---

combinations of several ideas. A complex program may contain numerous established concepts and algorithms as well as a multitude of innovative ideas. Whether a software program is a good one does not generally depend as much on the newness of each specific technique, but instead depends on how well these are incorporated into the unique combination of known algorithms and methods. Patents simply should not protect such a technology.

United States Patent and Trademark Office, *supra* note 6, at 24 (statement of Jerry Baker, Senior Vice-President, Oracle Corporation). Another software company executive described the potential impacts of software patents in similar terms:

> Creation of software will . . . be impeded by the difficulty of writing software that doesn't inadvertently trip across a patent somewhere. That is true in other fields where patenting is less controversial, but it's far worse in software. It's not unusual for a program to be a million lines long and consist of many thousands of subroutines and functions. Algorithms and ideas are embodied in each of those components and in combinations of them. Some of these algorithms may be studied in school or found in books, but many are developed "on the fly" as the program is created. Many of these subroutines and functions might be far afield from the purpose of the program as a whole.
>
> An operating system, for instance, might contain routines for sorting and searching, handling queues, parsing text, controlling hardware, testing memory, et cetera. It will be impossible to know which of these routines, algorithms and ideas violate a patent, because every programmer would need to understand every software patent—every software patent that is active. Software is simply too complex, composed of too many pieces which are too easy to create, to lend itself to being broken down into patent-sized chunks.

Id. at 46 (statement of Jerry Fiddler, Chairman of Wind River Systems, whose company creates software for embedded systems such as the microprocessor systems found in cars or phones).

compared against outstanding patents. Programming elements that are either copies of long used programming techniques or slight variations from long used programming techniques are unlikely to be covered by valid patents.[208] Even if patents have been obtained for these types of programming techniques, the patents are probably invalid.[209] In light of the long usage of the same or similar techniques, the techniques in question were either not new when the patents were sought or were merely obvious, unpatentable variations on the prior techniques in the field. In either case, patents on these sorts of techniques are void and unenforceable.

Two sorts of software innovations may be the subject of valid patents and should therefore be the targets of reasonable patent searches if these innovations are planned to be used in a new product. New software features that appear to be major departures from current programming techniques should be scrutinized to see if someone other than the party intending to use the features has independently developed and patented those techniques. If so, the patents involved may be valid since it appears that the programming techniques are new and non-obvious in comparison with other present programming techniques.[210]

In addition to presently non-obvious software advances, programming techniques that were non-obvious when they were originated may have been patented at that time. The resulting patent may be valid even though the techniques involved are

---

[208] *See* 35 U.S.C. §§ 102(a), 103 (1994) (stating that patents are not available for product or process designs that are either duplicates of, or obvious extensions of prior designs).

[209] *See* Amazon.com, Inc. v. BarnesandNoble.com, Inc., 239 F.3d 1343, 1351–66 (Fed. Cir. 2001) (questioning the validity and enforceability of a patent covering an on–line purchasing method where prior knowledge in the field suggested that the patented method was merely an obvious combination of known design features); Richards v. Chase Elevator Co., 158 U.S. 299, 301 (1895) (stating that patents may be declared invalid for lack of novelty).

[210] If no other patent is found that covers the new programming techniques and these techniques appear to be significant departures from earlier methods, then the developer planning to use the techniques may wish to consider filing for his or her own patents on those techniques. Assuming that the developer is the first to discover and disclose the new techniques and that the techniques are found to be non-obvious advances over prior practices, these new techniques will probably constitute patentable subject matter and warrant a patent for the developer. *See* 35 U.S.C. § 103(a) (Supp. V 2000); *see also* Graham v. John Deere Co. 383 U.S. 1, 13–19 (1966) (setting out the basic test for non-obviousness).

now commonly used. These sorts of formerly non-obvious departures from prior programming should therefore be targets of patent searches if the techniques are planned to be used in a new software product. Patents obtained when these techniques were new may still be in force, creating restrictions on the present use of the techniques. These restrictions may last for the life of a patent, which under current standards is 20 years from the patent application date and which, until recently, was 17 years from the date of patent issuance.[211] Patents obtained recently enough to be still enforceable under these standards and that cover innovations that were non-obvious when made should be targeted in software patent searches.

Two features of software development may make most of these older, still valid patents easy to detect. First, since most software development—in earlier periods as now—was incremental not revolutionary, few advances will have entailed the significant innovation over prior practices that would support an enforceable patent. Second, for those few advances that were non-obvious when made, patents obtained for these advances, the patents have probably figured in litigation that has gained some news coverage and notoriety, at least within the relevant software design field. These relatively notorious patents should be either known to software innovators working on products like the patented software or easy to find from published accounts of the corresponding litigation. The absence of enforcement actions concerning a particular type of technique that is now widely used will suggest, although not conclusively confirm, that there are no outstanding, valid patents from an earlier era covering that technique. With strategies based on these methods for limiting the types of patents that should be of concern to developers of new software, reasonably complete searches for software patents should be possible.

Problems stemming from unclear descriptions of software innovations in patent documents and corresponding weaknesses in patent indexing and searching may produce some errors in screening proposed software against existing patents, but it is unclear that these problems are any more significant for software advances than for other types of complex technological innovations. Even assuming, however, some difficulty or

---

[211] *See* 35 U.S.C. § 154(c)(1) (1994).

variation in how software advances are described, other contents of software patents may be sufficient to lead searchers to patents that will constrain a particular software design.

A patent covering a software feature must include descriptions of functional characteristics and applications of the feature.[212] These functional and application descriptions will give searchers workable targets for later detection of the patents involved. As they develop software with particular functional characteristics, developers should be able to search for patents covering prior innovations that produced the same functionality. If patents covering innovations with similar functionality are found, these patents can be given further scrutiny to determine if the functionality was implemented in the same way as in the proposed software. If so, the proposed software may infringe the patent. If a different implementation was used to achieve the same functionality, then it is unlikely that the new software will infringe the patent.

In any case, a search focusing on functional results of software designs should surface patents on the various designs that have been used to solve particular functional problems or to achieve particular functionality. Provided that software developers are at least clear and consistent in describing the functional results and advantages of their innovations—even if they vary in how they describe the inner details of the innovations—reasonably complete software patent searches should be possible. Searchers for patents governing a new software product or feature will have a reasonable chance of success if they take a functional approach to targeting their searches. By conducting initial searches for patents related to the particular functionality of the proposed new software or feature and then shifting to detailed scrutiny of the group of patents found in the initial search, developers should have a reasonable opportunity to detect and avoid patents potentially restricting their new software.

---

[212] *See* 35 U.S.C. § 112 (1994). The code states that the specification portion of a patent:

> shall contain a written description of the invention, and 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, or with which it is most nearly connected, to make and use the same.

Id.

3. Problems Stemming from Improperly Issued Patents

Software patents may also be undesirable if large numbers of invalid software patents are mistakenly issued by the Patent and Trademark Office (PTO). Due to weaknesses in the software-related prior art[213] information available to the PTO's patent examiners or weaknesses in the examiners' analyses of patent applications in relation to that prior art information,[214] invalid software patents may mistakenly be issued for either old software designs that have been in use for some time or for designs that are obvious, unpatentable variations on past designs. The threat of enforcement of such improperly issued patents would restrict the development of related software, at least until the validity of the patents could be challenged effectively.[215]

Concern over improperly issued software patents which do not meet patent law standards for novelty and non-obviousness stems from doubts about the ability of the PTO's patent examiners to recognize and reject patent applications for old or obvious software designs. There are two problems underlying this concern. The first relates to limitations on patent examiners' expertise concerning computer programming techniques and sources of prior art information.[216] While at one time the PTO did not recruit examiners with special expertise in computer science and related disciplines, this policy has been reversed and there is no reason to believe that the PTO will be any less able to establish a technically able body of examiners concerning computer software advances than in any other area.

---

[213] *See* Mohasco Indus., Inc. v. E.T. Barwick Mills, Inc., 221 F. Supp. 191, 195 (N.D. Ga. 1963) (defining prior art as "all patents, publications and public uses which have been in existence prior to a patentee's date of invention or more than one year prior to his filing date).

[214] *See* Cantzler, *supra* note 127, at 456 (stating that examiners need to have backgrounds in computer science, a requirement that was, until recently, overlooked by the PTO as a patent attorney qualification).

[215] Obtaining a patent is a costly and time consuming process. Even if there are no problems, it can take 18 months to three years from patent application to the patent grant. If problems develop, the process can extend to over five years. Prosecuting a patent that is not complex costs approximately $5,000 to $20,000, but the costs can increase if problems develop. *See* Thomas B. Burke, *Software Patent Protections: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1117 n.12 (1994).

[216] *See, e.g., United States Patent and Trademark Office*, *supra* note 6, at 27 (statement of Kaye Caldwell, President, Software Entrepreneurs Forum) (describing the need for patent examiners who have computer programming expertise).

A second reason why patent examiners may be less effective in reviewing software patent applications than in other technological areas may be harder to overcome. Reviews of software patent applications are hindered by large gaps in prior art records that are peculiar to records of software development.[217] This problem is due, in part, to earlier confusion about the patentability of software which caused a great deal of innovative software not to be patented and therefore not to be recorded in patent records.[218] Unlike the case in other technological fields where strings of patents tend to record the major innovations and trends in the fields, many aspects of advancing software technology are not disclosed in published patents.

This would appear to be a temporary problem, however. This gap in software technology records will be resolved as software patents and associated invention disclosures become more common. To the extent that sources of information available to examiners about past software designs are more limited than comparable prior art information in other technological areas, it may be useful to provide expedited means to challenge the validity of software patents soon after those patents are issued. These expedited procedures should include opportunities for interested parties to submit prior art information to fill gaps in the software patent record.

For example, at least until the overall body of software patent records improves, a procedure providing for the regular reexamination of software patent validity early in the life of an issued software patent may be desirable. Such a procedure would ensure that parties threatened with infringement liability under the patent—and possessing a corresponding motivation to identify and disclose invalidating prior art information—would have an opportunity to have the validity of a suspect patent tested in a less expensive manner than through litigation challenging the patent. Invalidation of patents under this type of procedure would remove improperly issued software patents

---

[217] *See id.*; *see also id.* at 73 (statement of Richard Stallman, Free Software Foundation) (describing problems with PTO examiner's evaluations of the obviousness of software advances); Cantzler, *supra* note 127, at 456–57 (describing the problems arising from gaps in prior art records concerning software).

[218] *See generally* Alan P. Klein, *Software Patenting: A New Approach*, 6 U. BALT. INTELL. PROP. L.J. 135 (1998) (discussing cases that have examined the PTO's treatment of patent applications and what constitutes prior art).

before the threat of enforcement of those patents had a substantial opportunity to deter and diminish subsequent software development. Presently, the initiation of a reexamination depends on a discretionary decision of the Commissioner of Patents to begin such a proceeding. The Commissioner has indicated a willingness to entertain this type of patent challenge through petitions seeking the reexamination and potential invalidation of issued software patents.[219] Few parties have sought such examinations of software patents to date,[220] in part because there is no guarantee that the PTO will even conduct a reexamination review of an issued patent based on submissions of previously unaddressed prior art information. A better practice would be to schedule an automatic reexamination of software patents at a reasonable time (for example, six months) after the patents issue.

CONCLUSION

Until recently, uncertainty about the scope of software patent protections caused the computer industry to develop without either the incentives or the restrictions of software patents. Changes in the industry suggest that there is an increasing need for software patent incentives to promote future software development. In particular, the increased concentration of economic and marketing power in a few large computer industry firms has placed a new premium on obtaining software patents. Software patents promise to serve important roles in protecting future software innovations by both large and small–scale software developers.

For small–scale developers, a software patent may be the only effective way to develop and market an innovative new form of software without having key features of the software scooped up by large competitors with no compensation to the small

---

[219] *See, e.g.*, Q. Todd Dickinson, Reconciling Research and the Patent System, ISSUES IN SCIENCE & TECHNOLOGY, July 1, 2000, 2000 WL 20687129 (comments by Commissioner of Patents) (noting a willingness to initiate discretionary reexaminations of patents "when the prior art and broad public concern warrant it").

[220] *Id.* (comments by Commissioner of Patents) (noting that "surprisingly few" parties initiate reexaminations of patents); *United States Patent and Trademark Office*, *supra* note 6, at 59 (statement of Lee Patch) (describing the general apprehension toward the reexamination process among members of the software industry).

innovator. The superior marketing abilities and product integration advantages that large concerns possess relative to small innovators will tend to cause larger concerns to win out over smaller ones in head to head marketing of products incorporating the same sorts of innovative features. Absent patent protections, small concerns can hardly hope to win such a battle. Furthermore, absent some hope of winning such a contest, a small innovator may see little chance to profit from a complex software innovation and therefore forgo the development or marketing of innovative software entirely.

Venture capitalists or business partners who might give financial backing to the efforts of these small innovators may withhold necessary investments on similar grounds. The potential impact of software patents in securing financial backing for small software developers is reflected in the experience of MacinTax, a small startup company founded by Mike and Susan Morgan. Because the Morgans obtained patents on their software products, they were able to protect themselves and their investors from the appropriation of their innovations by other firms. The reassurance that patent protections provided was critical in attracting investment in the company by venture capitalists. As described by Susan Morgan,

> [W]hen venture capitalists asked us how we could protect ourselves against say Microsoft coming out with a competitive product and stealing our market, the fact that we had applied for patents put the problem to bed. It made the [venture capitalists] feel much more comfortable, and that's a big difference.[221]

For large concerns, software patents may serve a more defensive purpose. As innovators within such firms develop new software designs, these firms may seek related patents on the innovations to ensure that they retain control over the new designs. If they do not, they may find that they are restricted by another company or party that independently develops a similar software innovation and is the first to obtain a related patent. A software patent obtained by a large, innovative concern may be the best way to establish that company's early development of a given technology and to prevent other parties from imposing

---

[221] *United States Patent and Trademark Office*, *supra* note 6, at 12 (quoted in statement of Paul Heckel).

patent restrictions on the same technology. This sort of race by large companies to obtain patents on software advances is likely to produce earlier disclosures of innovative software in publicly available patents than would be the case without patent incentives for disclosure.

Of course, these disclosures are made in exchange for a period of patent restrictions on the inventions disclosed. Some of the technology disclosed may not be claimed in the related patents and therefore enter the public domain immediately. In addition, the software designs that are claimed will often be available for licensing from the patent holder, resulting in immediate access by companies and consumers to software designs that might otherwise have remained undiscovered or undisclosed for some time.

Software patents may also serve a different defensive function for large concerns by giving those concerns additional bargaining power in the face of patent infringement claims by other software producers. Where large software producers develop software design expertise in different areas, each may obtain patents in their respective areas of specialization. In order to adopt the best design features developed by multiple large companies, a large software producer may need to gain permission under patent licenses from several large companies. Software patent cross-licenses can provide a means to obtain this permission at little or no cost to the licensee. Under these agreements, each patent holder licenses the other party to use the invention features covered by their respective patents.

In some cases, this cross licensing will occur as new products are designed and competitors learn that they and their competitors each have produced different specialized discoveries and patents. These parties may voluntarily cross license their patents because each realizes that they will benefit from the resulting access to the full range of advances in the field.

Alternatively, cross licensing may occur through litigation in which the patent rights of one competitor are used to compel access to the technology controlled by another. By raising the threat of an infringement action, but offering to settle or avoid the litigation through entry into a cross licensing agreement, a firm can use its patent portfolio to force other concerns to provide access to valuable technologies. This access may be gained with little or no revenue outlay other than the cross licensing of the

benefited firm's patents. An official for Oracle Corporation described this type of use of a patent portfolio as follows:

> Oracle has expended substantial money and effort to protect itself by selectively applying for patents which will present the best opportunities for cross-licensing between Oracle and other companies who may allege patent infringement. If such a claimant is also a software developer and marketer, we would hope to be able to use our pending patent applications to cross-license and continue our business unchanged.[222]

For both small and large-scale software developers, clearer standards for granting software patents should provide stronger incentives for creating innovative software. Beyond their impact in producing particular software products, software patent rewards should produce useful distinctions between innovators and non-innovators in times of intense industry competition. Software patents will promote these distinctions by enhancing the economic strength and financing opportunities of software innovators relative to their less innovative competitors. In highly competitive markets, this will help the innovators to survive and the less innovative companies to be displaced by more innovative ones.

Finally, patent controls that encourage companies to develop and market new software with highly original designs should expand the number and diversity of innovative software products. As more numerous and varied software products are made available, the functional benefits of these software products should flow to the public. Patents encourage the diversification of technological development in many fields. Patent incentives for software development can play a similar role. These important incentives and the public benefits resulting from the diversification of software development should be embraced through strong support for software patents and the better living innovative software promises to produce.

---

[222] *United States Patent and Trademark Office*, *supra* note 6, at 24 (statement of Jerry Baker, Senior Vice-President, Oracle Corporation).