Technology and Innovation, Vol. 19, pp. 787-798, 2018

ISSN 1949-8241 • E-ISSN 1949-825X
http://dx.doi.org/10.21300/19.4.2018.787
www.technologyandinnovation.org

ON THE SOFTWARE PATENTING CONTROVERSY

Neil G. Siegel[1] and Marek A. Suchenek[2]

[1]Daniel Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA, USA
[2]Department of Computer Science, California State University Dominguez Hills, Carson, CA, USA

This article discusses software patenting, beginning with a brief history and a description of the legal foundations for software patenting, and then considers major arguments pro and con regarding software patenting. It advocates a thesis that the economic, political, and societal benefits of software patenting tend to outweigh its costs and negative effects, even while acknowledging that the software patenting process is arguably imperfect.

Key words: Software patents; Business implications of software patents; History of software patents; Legal environment for software patents

INTRODUCTION

The question of software patentability has been hotly debated and a subject of considerable controversy over the last few decades. Over the years, the courts have weighed in on that question with evolving, if not contradictory, interpretations of the noun "process" (defined in Title 35 U.S. Constitution § 101, Inventions patentable) in the context of its patentability, inventing new tests and criteria that, perhaps, clarified some relevant issues but left others more complicated and confused than they were before.

Let us begin with with a few definitions (1):

- An *algorithm* is "a step-by-step procedure for solving a problem or accomplishing some end"
- *Software is* "the entire set of programs, procedures, and related documentation associated with a mechanical or electronic system, and especially a computer system"
- *Code* is "instructions for a computer (as within a piece of software)"
- *A mathematical formula* is "a general fact, rule, or principle expressed in mathematical symbols"

That is, software is the entire set of code that comprises the instructions for a computer for some purpose. Code is a segment of such software: an individual line or multiple such lines. And an algorithm is a description of the sequence of key steps to be used; this, however, can be very complicated. A mathematical formula is a more abstract statement than an algorithm; a formula can be considered a general fact or statement, whereas an algorithm has the potential to be far more situationally specific. Formulas and algorithms need not be in a form that is understandable to the computer, nor need they embody every step and condition to accomplish a goal; instead, they

Accepted: March 1, 2018.
Address correspondence to Neil G. Siegel, Ph.D. (member, NAE and NAI), The IBM Professor of Engineering Management, Daniel Epstein Department of Industrial and Systems Engineering, Viterbi School of Engineering, University of Southern California, 3650 McClintock Avenue, OHE 310D, Los Angeles, California 90089-1450, USA. Tel: 310-375-9907. E-mail: nsiegel@usc.edu or siegel.neil@gmail.com. Website: www.neilsiegel.usc.edu

may focus only on the most important of such steps; code and software, by definition, must be in a form that is understandable to the computer and must be complete.

In the early days of computer technology, the U.S. Patent and Trademark Office (USPTO) routinely refused patents on inventions with software components. This informal policy was formalized into regulation in 1968 when the USPTO issued new guidelines in which computer programs were expressly declared to be unpatentable.

In 1998, the U.S. Court of Appeals for the Federal Circuit in the case State Street Bank and Trust Co. v. Signature Financial Group ruled that if software yielded "a useful, concrete, and tangible result," then it should be considered patentable. This ruling greatly facilitated patenting of computer software. But, at the same time, it invigorated those opposed to the idea.

The year 2007 marked the beginning of a reversal of the trend that made many software patents possible. The U.S. Court of Appeals for the Federal Circuit said in an opinion that if the invention consisted of a process that was entirely mental, then it was unpatentable even if it was augmented with a modern electronic device such as a computer. In what later became a highly-publicized case, a patent examiner rejected the patent application for an algorithm that minimized risks in commodities trading "on the grounds that the invention is not implemented on a specific apparatus, merely manipulates an abstract idea, and solves a purely mathematical problem." That decision was challenged. In 2008, the U.S. Court of Appeals for the Federal Circuit in the case In re Bilski upheld that rejection, based on the so-called machine-or-transformation test for patentability, which, in the Court's opinion, the algorithm did not pass. Since most computer programs were also likely to fail that test, many considered the Court's ruling the last nail in the coffin of software patents.

However, in the 2010 case of Bilski v. Kappos, the U.S. Supreme Court partially reversed the Federal Circuit's decision and rejected the machine-or-transformation test as the sole criterion of patentability. Although the Supreme Court still ruled against granting of a patent for the said invention, the decision was not against general patentability of any algorithm or a business process that failed the said test but was based instead on the fact that a patent for the invention in question would preempt an abstract idea.

In this paper, we will review some arguments for and against software patents.

THE CURRENT LAW

Article I, Section 8 [8] of the U.S. Constitution (Powers of Congress) states: "The Congress shall have Power […] 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."

This provides the Constitutional basis for the U.S. patent law (Title 35 U.S.C. § 101, Inventions patentable): "Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title." § 100 of that Law defines "process" as "process, art or method," including "a new use of a known process, machine, manufacture, composition of matter, or material."

Since a computer program (or an algorithm) clearly falls into category "process" so defined, one could argue that in light of the patent law, software should be patentable. But the question of software patentability turned out to be much more complicated than the Section 8 [8] and § 101 seem to suggest, and for the last few decades has been hotly debated and a subject of considerable controversy. Over the years, the courts have weighed in on that question with evolving, if not contradictory, interpretations of the noun "process" in the context of its patentability, inventing new tests and criteria that, perhaps, clarified some relevant issues but left others more complicated and confused than they were before.

THE CONTROVERSY

In the early days of computer technology, the USPTO routinely refused patents on inventions with software components. This informal policy was formalized into regulation in 1968 when the USPTO issued new guidelines in which computer programs were expressly declared to be unpatentable.

Software-related patents became a controversial issue after the U.S. Supreme Court decision in the 1981 case of Diamond v. Diehr opened a narrow

possibility for patents related to software, so long as they were parts of otherwise patentable inventions or processes. In that case, the Court ordered the USPTO to grant a patent to a method for curing rubber, in which the only use of software was to time the heating process. As a result, many software programs have since been granted patent protection, including the RSA public-key cryptography algorithm in 1983, the Lempel-Ziv-Welch data compression algorithm in 1985, and Karmarkar's linear programming algorithm in 1988. In turn, a number of individuals (some of them prominent scientists) and organizations launched their advocacy against software patents.

In 1998, the U.S. Court of Appeals for the Federal Circuit in the case State Street Bank and Trust Co. v. Signature Financial Group ruled that if software yielded "a useful, concrete, and tangible result," then it should be considered patentable. This ruling greatly facilitated patenting of computer software. But, at the same time, it invigorated those opposed to the idea. The emergence and proliferation of so-called "patent trolls"— usually, law firms that specialized in buying a wide variety of patents solely for the purpose of subsequently suing manufacturers for patent infringements — also fueled growing discontent with the legal post-1998 status quo.

The year 2007 marked the beginning of a reversal of the trend that made many software patents possible. The U.S. Court of Appeals for the Federal Circuit said in an opinion that if the invention consisted of a process that was entirely mental, then it was unpatentable even if it was augmented with a modern electronic device such as a computer. In what later became a highly-publicized case, a patent examiner rejected the patent application for an algorithm that minimized risks in commodities trading "on the grounds that the invention is not implemented on a specific apparatus, merely manipulates an abstract idea, and solves a purely mathematical problem." That decision was challenged. In 2008, the U.S. Court of Appeals for the Federal Circuit in the case In re Bilski (the source of the previous quote) upheld that rejection, based on the so-called machine-or-transformation test for patentability, which, in the Court's opinion, the algorithm did not pass. Since most computer programs were likely to fail that test too, the court's ruling was considered by many the last nail in the coffin of software patents.

The machine-or-transformation test, according to Stefania Fusco, is defined as follows:

> In United States patent law, the machine-or-transformation test is a test of patent eligibility under which a claim to a process qualifies to be considered for patenting if it 1) is implemented with a particular machine, that is, one specifically devised and adapted to carry out the process in a way that is not concededly conventional and is not trivial; or else 2) transforms an article from one thing or state to another. (2)

However, in 2010, the U.S. Supreme Court, in the case of Bilski v. Kappos, partially reversed the Federal Circuit's decision and rejected the machine-or-transformation test as the sole criterion of patentability. Although the Supreme Court still ruled against the granting of a patent for the said invention, the decision was not against the general patentability of any algorithm or a business process that failed the said test but was based instead on the fact that a patent for the invention in question would preempt an abstract idea (this phrasing is a standard criterion of non-patentability of an invention). As a result of these two rulings (in 2008 and 2010), the USPTO pursued a policy of refusing a patent to any computer program (referred to as a "method") that failed the machine-or-transformation test "unless there is a clear indication that the method is not drawn to an abstract idea." This has been a more stringent criterion than the "useful, concrete, and tangible result" test, thus making software patenting substantially more difficult than it was in the years 1998 to 2007.

Nevertheless, the Supreme Court was careful to not comment on software patentability, solely pointing out the difficulties that the Information Age raises for the patent law. Wrote Justice Kennedy:

> It is important to emphasize that the Court today is not commenting on the patentability of any particular invention, let alone holding that any of the above-mentioned technologies from the Information Age should or should not receive patent protection. […] [T]he patent law faces a great challenge in striking the balance between protecting inventors and not granting monopolies over

> procedures that others would discover by independent, creative application of general principles.

In Alice Corp v. CLS Bank International, the Supreme Court (3) made a clearer statement that an abstract idea, even if implemented on a computer, could not be patented. The decision does not mention either the words "software" or "algorithm" (4), but the decision was "widely considered as a decision of software patents for business methods" (5). For example, a Federal Court from the Northern District of California stated:

> By clarifying that the addition of a generic computer was not enough for . . . patentability, Alice has had a significant impact on software patents. In Alice's wake, the Federal Circuit and numerous district courts have wrestled with the issue of whether various software patents disclose the "inventive concept" required for patentability. (9)

In addition to the judiciary, the legislative and executive branches of the Federal Government have interested themselves in the question of what software, if any, merits patent protection. Within the executive branch, activity has centered on the USPTO, which establishes and executes the procedures for obtaining patents (6). According to Orozco, "the (US)PTO launched a Business Method Patent Initiative, which included industry outreach and quality programs," and took other steps to investigate whether computer-implemented business method patents encouraged or curbed growth in innovation (7).

The legislature has both held hearings and passed legislation that relates to this subject. Business method patents have been the subject of many hearings, for example, a hearing in 2001 was dedicated to this subject (8). According to Orozco, the "first instance of legislation targeting business methods was The American Inventors Protection Act of 1999. This legislation modified the Patent Act to provide a first user defense against business methods" (7).

The recently passed overhaul (The America Invents Act, H.R. 1249, signed into the law September 16, 2011) of the patent law did not, however, directly address the patentability of software.

In this paper, we will review some arguments for and against software patents.

CONSTITUTIONAL AND LEGAL ARGUMENTS

The recurring legal objection against patenting software that the courts routinely articulate is a concern that a patent, if granted, would pre-empt a mathematical fact or an abstract idea (it has been a widely-accepted legal doctrine that laws of nature, physical phenomena, and abstract ideas are non-patentable), thus restricting others in their independent searches for new inventions and discoveries. This concern with potential pre-emption has been widely reiterated by the opponents of software patents. What was conspicuously missing in arguments of this sort was a clear and commonly accepted definition of "mathematical" and "abstract." In the struggle to sort this out, various tests were invented and applied. For instance, one posits whether "a set of numbers is computed from a different set of numbers by merely performing a series of mathematical computations"; if answered affirmatively, this would yield a conclusion that the algorithm in question is "mathematical" and, hence, not patentable. In 1989, the Office of the Solicitor of the USPTO released a legal analysis that attempted to define a concept of "mathematical algorithm" in the context of patentability.

Aside from the fact that it is difficult to find in patent law the specific objection against the patentability of a mathematical algorithm, the above-mentioned efforts of courts and lawyers to differentiate between the "mathematical" and "non-mathematical" algorithms have been summarily criticized as inadequate or absurd, even by some opponents of software patenting. Courts did offer an opinion in this matter. According to an analysis by Associate Solicitor Lee E. Barrett, an attorney in the Office of the Solicitor of the USPTO:

> The Supreme Court thus recognizes that mathematical algorithms are the basic tools of scientific and technological work, and should not be the subject of exclusive rights, whereas technological application of scientific principles and mathematical algorithms furthers the constitutional purpose of promoting 'the Progress of … Useful arts' (18). It is also recognized that mathematical algorithms may be the most precise way to describe the invention.

The quoted analysis also acknowledges that "[l]egislative history indicates that Congress contemplated that the subject matter provisions be given a broad construction and were intended to 'include anything under the sun that is made by man'" (9).

In some cases, algorithms have been thrown into the same category as mathematical formulas, as in Diamond v. Diehr, which produced this rather controversial opinion: "[A]n algorithm, or mathematical formula, is like a law of nature, which cannot be the subject of a patent." This seemed to be a result of the belief held by some judges that everything that can be precisely defined is somehow equivalent to a mathematical formula (its definition) and — therefore — unpatentable; this view has also been advocated by Ben Klemens (10).

Equating algorithms with mathematical formulas is difficult to justify. For example, try to find a mathematical formula for the function f computed by this Java code:

```
public static int
f(int n)
{
                    if (n<=1) return 1;
                    if (n%2==0)
                    return (f(n/2) + 1);
                    else return (f(3*n + 1) +1);
};
```

For some seven decades now, mathematicians and computer scientists have not been able to figure out if this program halts for every integer n or not. This illustrates that software code and algorithms do not always correspond in a simple and obvious fashion.

Not all courts subscribed to this idea. For instance, Justice Stone wrote for the U.S. Supreme Court in Mackay Radio & Telegraph Co. v. Radio Corp. of America: "While a scientific truth, or the mathematical expression of it, is not a patentable invention, a novel and useful structure created with the aid and knowledge of scientific truth may be."

As noted above, when we provided definitions, a mathematical formula is a more abstract statement than an algorithm; a formula can be considered a general fact or statement, whereas an algorithm has the potential to be far more situationally specific. This is, in our view, a critical opportunity to achieve a definition of patentability: When an algorithm (and the software code that implements it) is elaborated to the point of becoming situationally specific, that may be a sign that the work has risen to a level that warrants patent protection. For example, the use of an established scientific principle (which might be described by a mathematical formula), such as Boyle's Law or Ohm's Law (both defined in (11)), even if described in an algorithm that is more detailed than the mathematical formula, likely would not warrant patent protection, but by adapting such a scientific principle to a specific situation in a fashion that results in being able to accomplish a useful result not previously achieved through that established scientific insight, it might rise to a level of innovation that warrants patent protection even if that innovation is described using a algorithm.

One could extrapolate the situation in Alice Corp v. CLS Bank International along these lines. According to Nazer and Ranieri, Alice Corp.'s patent simply claimed a form a escrowing — called an intermediate settlement — that was well-known (4). Their patent simply implemented this known art and method into software instructions, but it did not accomplish something that was either new, more accurate, or provided some otherwise novel capability; it just did what was already known, but did it through the use of a computer. Use of a computer does in itself provide some benefits — it might be faster or more consistent, etc., but those benefits results from the nature of the computer and not from the nature of the process of intermediate settlement.

Until the courts adopt some specific guidance, whether along the lines of that which we indicate above or something else, it remains unclear what is a patentable matter in the court's interpretation of the patent law.

ECONOMIC ARGUMENTS

The economic argument for patent protection of software parallels that for intellectual property in general: Innovation is good for society, so it is appropriate for the government to take actions that promote such innovation. In the U.S., patents are one of the cardinal approaches for providing such promotion of innovation, attempting to create an economic incentive by granting a period of monopoly to the inventor. As noted above, the authority to do so actually resides in the U.S. Constitution.

Monopoly as a method of economic incentive, and also a means to provide a nurturing period of operation without competition, has long existed in the English legal system. English sovereigns have long granted patents that gave exclusive right to a single English company to trade with a specific foreign country; the English East India Company is probably the most famous example.

There is also a long history of people finding economic value in information, whether in the form of military secrets, in the form of industrial trade secrets (the Chinese worked very hard for hundreds of years to keep their methods for the production of both silk and porcelain unknown to their buyers), and in other forms. Merchants have for centuries voted with their pocketbooks that information has true value.

Some economists (for instance, Kenneth Arrow (12)) argue that information has become a commodity that should be traded on a free market. They tend to see patents as market instruments that make such trading practically possible and, therefore, have a positive benefit to society. In order to sell a commodity (say, for instance, a barrel of crude oil), the seller has to have a monopoly (the exclusive right, if you will) on a particular instance of that commodity, for otherwise the prospective buyer will not have a reason to pay him for it. With information, the trading becomes tricky: Unlike in the case of crude oil, once the seller has disclosed the information to the prospective buyer, he loses the monopoly on it, and, therefore, the potential buyer has no incentive to pay for it. Moreover, the buyer may pass it for free to other prospective buyers, thus depriving the seller from the proceeds of future sales of the information in question.

Despite this difficulty, we believe that this view of information as a commodity sheds light on the software patenting controversy.

The code of computer software certainly carries information (some would say *knowledge*): what decisions and what actions and in what order those decisions and actions must be made in order to accomplish the desired computational result. Copyright protects a particular expression of that information but not necessarily its *contents* or the underlying knowledge, One can use a copyright to protect your recording of a song that was composed by someone else; at the same time, other people can have copyrights on *their* recordings of the same song. So, if information and knowledge (not just an expression thereof) can be traded, then there seems to be no good reason why software could not. In the existing legal system, a patent is the only practical means of protection that allows for trading the information and knowledge carried by computer software. As much as it may lag behind the needs of the Information Age, patent law attempts to minimize the difficulty of creating a market for software. The emergence of patent law is likely to boost the supply of high-quality software and, thereby, provides a benefit to society.

So, from the perspective of information theory, it does not make much economic sense to end software patents.

The economic perspective provides a means to counter some arguments articulated against software patents. A common argument in this category asserts that the inventor or creator of the software in question is not deprived of its use or other benefits just because other people are using it. But such an assertion is based on a fallacy that ignores the fact that the value of information depends not only on its contents but also on a lack of knowledge of it by others, including prospective buyers (in economics, this subjective dependence is referred to as asymmetry).

Let's consider as an example a game of poker. Imagine yourself as a player. When you look at your poker hand, the information and the knowledge that you acquire this way has a very real monetary value to you. For instance, if you have a strong hand, you may bet high; if, however, your hand is weak, you may consider folding, thus avoiding almost certain financial loss. You could not make a rational choice between the two actions if you did not know what your poker hand contained.

Now, suppose one of players wants to see your hand for his obvious benefit. He can call, a move which requires him to risk money that may end up in your pocket (if it turns out that you have the strongest hand). It is as if he offered you a patent (copyright would not accomplish the same thing, as a simple rearrangement of cards in your hand would become a different expression of the same information) for your poker hand. You may consider whether the value that he offers you this way is enough to show

your hand. If it is not, then you can raise and keep playing instead.

But what if one of the players demanded that you just show your poker hand for free? He might argue that by showing your hand to others you do not lose any use of the cards that you have in your hand. If you accept such an argument, we would love to play poker with you.

The above example illustrates fairly well the fallacy that some of the staunch opponents of software patents commit: They ignore what is called an asymmetry in information and information trading. A software patent allows the inventor to benefit materially from disclosing the invention in public, and the general public benefits by learning, for a fee, from the disclosure how to make a particular product or decision.

Another economic argument that is often brought up by the opponents of software patents is an assertion that the patents yield little, if any, economic benefit for the general public. In particular, it is argued, there is no convincing evidence that software patents spur any substantial innovation or progress in the creation of new software. Some argue that the inventors of new software algorithms would invent them anyway, whether rewarded with patents or not.

For instance, Gregory Clark has assembled evidence to try to show that institutional means of protecting property (including means of protecting intellectual property) do not in general spur economic progress (13). Those who make these arguments generally conclude that these protections were not in fact effective at stimulating *new* economic progress but conclude that they were effective at sustaining and nurturing those innovations that were already extant. Such an argument would, therefore, actually demonstrate the value of software patents. If you accept these arguments, the value is realized through such sustainment rather than through the creation of new innovations, but that is still a value for society.

Some economists draw a different distinction, asserting that a patent is an incentive to *disclose* an invention rather than an incentive to create it. Clark reports that "[t]he establishment of […] patent rights in northern Europe in the sixteenth century arose from the desire of countries to attract foreign artisans […] [who] would not emigrate without legal guarantees that their knowledge would be protected." This is a different insight into the potential rationale for intellectual property patents but still would seem to provide a mechanism by which society can benefit from the software patent process. Society benefits by the disclosure (which shares knowledge that can be built upon by future inventors), while the inventor receives the potential of an economic reward (in the form of a period of monopoly use of his/her invention) in return for this disclosure.

This line of reasoning asserts that software patenting does protect the interest of the authors and inventors while providing them with incentives to disclose the source codes of their programs to the public, which is clearly a benefit for the society at large. The (relatively poor) economic track record of those nations that have *not* protected intellectual property would seem to provide supporting evidence for the proposition that such a balance is of significant net value to a society.

We see this incentive-based approach as consistent with the philosophy of government established by the U.S. Constitution — one where the role and authority of the government is limited. The government and the laws were not established in order to correct or stimulate the behavior of the people but to protect law-abiding citizens from the occasional asocial elements. Therefore, an approach that gives control of the process to the inventor seems consistent with the principles of the U.S. Constitution. And, remember, there is always another approach available to the inventor (in the U.S. at least): the trade secret, wherein the mechanism by which a software effect is acheived is not disclosed but then is protected only to the extent that the inventor takes steps to prevent such disclosure. Note that, in this case, the society does not give the inventor the period of monopoly use but also does not receive the benefit of knowledge-accrual that results from disclosure. It is a true alternative. And the decision — appropriately, in our view — rests with the inventor and not with the government.

It is worth noting that there are examples in the real world that might be deemed to support the opposing position: that not allowing software patents is conducive to economic benefits to society. For

example, during the period that software patents were not allowed, or allowed only on a very limited basis, many seminal software concepts were created and brought to market: the full-screen word processor, the spreadsheet, the mouse-icon human interface, and others. It is perfectly reasonable to argue that the lack of patent protection for these seminal ideas allowed robust competition in the quality of the implementation, with the market eventually choosing winners and losers. For spreadsheets, for example, we remember successive market dominance by Visicalc, then Lotus 123, then Excel, with a whole host of other competitors along the way. Many will argue that this was good, but one might also argue that there were other reasons for this robust period of innovation (for example, it was just the start of the software revolution, a time when one might expect more new ideas to be emerging than in a more mature market) and that the lack of software patenting at this time was not a cause of the burst of innovation, but instead just a coincidence, and merely served to deprive the innovators of the rewards from their inventions.

If you were the original inventor of the computer spreadsheet, made little or no money from your idea, and saw others making a fortune, you might well believe that you had a convincing argument that, in this case, the creator of an important innovation was not rewarded in an appropriate manner, thus making him more likely to seek other means of subsistence than to work on his new inventions. You might wonder where the incentive was for the next innovator to spend time thinking up something that could have the same impact as the computer spreadsheet.

On the other hand, there are those who look at the large amount of software-patent litigation today and say that (a) software patents — and the ensuing legal fights — are diverting resources from the creation of new and better products, and that this is not good for society; and (b) the large software-patent portfolios wielded today by some companies have become significant obstacles to creation and innovation, as financial backers get nervous about the possibility of the start-up companies in which they might invest being sued by the companies with the large software patent portfolios. One can also point to the acquisition of companies primarily for their patent portfolios (some assert that this is why Google aquired the phone business formerly within Motorola) as a potential waste of resources. On the other hand, the big guys don't always win, as exemplified by Apple losing its patent protection for "pinch-to-zoom," when a court ruled that a 2005 patent on the same subject by a small company called Applied Minds pre-dated the Apple filing (as is explained in "Apple's pinch-to-zoom patent meets its second end" (9)).

In the long run, though, software patents, as well as other institutional and legal instruments of protection of intellectual and physical property rights, in our view, do appear to contribute to economic and technological progress. In the absence thereof, many successful inventors, innovators, and entrepreneurs might have not decided to pursue careers in the software industry, opting instead for other, more rewarding or profitable venues. Such absences must result, eventually, in slowing the rate of innovation, as well as productivity, in software development.

An interesting insight into the question of appropriateness of ownership of information that may help to sort out some controversies pertaining to software patenting was offered by federal Judge Richard Posner (14). He argued that in order not to discourage up-front investments in pursuit of information, useful information that was costly or difficult to acquire should allow for its ownership and protection, with the exception of information that prevents such nuisances as fraud, deception, or misrepresentation (for instance, an individual's criminal record and credit history), which should — by default — be left in the public domain and, therefore, be presumed non-proprietary. Although Posner was merely addressing certain privacy issues from an economic perspective, his ideas appear strikingly congruent with provisions of the patent law that require an element of difficulty or cost in a discovery of patentable invention and attempts to reconcile legitimate rights of the inventor with the needs of the society as a whole.

The public, now accustomed to cheap but powerful digital devices, has also apparently developed a sense of entitlement to free software, a sentiment that has been fed and nurtured by various anti-software patenting organizations and prominent individuals. But if the software were always free, then those who create it would have to live off of public generosity and the charity of philanthropists. The most likely

outcome is that, over time, much less software would be available, to the considerable detriment of society.

We conclude that the above show that software patents are, in fact, valuable to society; software is the major element of many critical societal systems (15), and society will benefit by motivating additional such innovations. There was a short period of innovation that occurred during a time when software patents were not allowed, but, in our assessment, this was a transient effect at the beginning of the software era. The level of innovation was due to that newness and not to the lack of software patenting.

POLITICAL, ETHICAL, AND PHILOSOPHICAL ARGUMENTS

It is worth remembering that for more than 2,000 years, legal and economic arguments were used to justify slavery (it is also interesting to note that some studies, for example Suzanne Miers' book *Slavery in the Twentieth Century,* offer evidence that there are more slaves in the world today than any time in human history). So one may wish to scrutinize the arguments for and against software patents from perspectives other than that of the law and economics as a precaution against unintentional but undesirable consequences. In this section, we therefore consider political, ethical, and philosophical arguments. Some authors, of course — Heinlein in his book *Starship Troopers* is a famous example — would consider this list redundant, arguing that politics is simply the embodiment of public ethics and philosophy.

The reason why societies provide protection for intellectual property is to stimulate future innovation through the positive example of allowing past innovators a chance to benefit from their innovations; the mechanism used to implement this chance to benefit is a period of monopoly, which is what a patent provides. In The Federalist Papers (the essays that in essence form the discussion and argument behind the U.S. Constitution), this approach was rationalized not on economic grounds, but on philosophic, moral, and ethical grounds — in essence, on political grounds. The founders were indeed concerned with "promoting the general welfare," but their primary concern — as we will establish below — was promoting the "natural rights" of mankind.

Historically, within the U.S., the argument for such protection of intellectual property is political rather than economic. It is a dual argument that says both that (a) some economic success associated with the creation of ideas is necessary to get new opinions out into the "marketplace of ideas" and that there is political benefit to the dissemination of a range of such ideas; and (most especially) (b) U.S. legal tradition displays a strong penchant for a philosophy of personal property ownership. Remember that in many of the societies that existed concurrent with the founding of the U.S., all of the property was owned only by the sovereign. No private person could own property; instead, they could only obtain something akin to a lease, which could be revoked by the sovereign at their pleasure, at any time, for any reason. This was felt by the founders of the U.S. to be an economic obstacle to prosperity, but even more so as a moral and ethical barrier to justice.

What brought about this attitude was a religious revolution. In particular, in England during the 17th century, there was a strong movement to revert back to the Old Testament — the Jewish Bible — for scriptural authority rather than the New Testament; this is the basis for many well-known religious movements of the time, including the Pilgrims and the Society of Friends. The philosophical emphasis in the Old Testament is on justice in this world (the book of Amos is a good exemplar of this philosophy) rather than the focus on individual salvation found in the New Testament. This reversion to a focus on the Old Testament colored the philosophical tendencies of the next 200 years. The U.S. founders all grew up steeped in that tradition; those who left England for the North American colonies tended to be disproportionately proponents and adherents of the new religious philosophy.

Furthermore, English politics of the time were overtly corrupt, with practices like buying votes, both from the populace in general elections, and from members of Parliament for votes of bills, being open, acknowledged, and pervasive. Several of the U.S. founders were disgusted by this and cited this as a reason all on its own to separate the Colonies from England; Benjamin Franklin, who lived in London during much of the decade prior to the U.S.

Declaration of Independence, was a particularly emphatic writer along these lines (see, for example, the writings of Franklin cited by Barbara Tuchman in her book *The First Salute*). Among the rationales offered in England for this corruption was that it was supportive of economic prosperity. The conclusion drawn by several of the U.S. founders was, therefore, that not every action that helped economic prosperity was a good idea.

The combination of factors cited above provides a rationale for believing that the focus in the U.S. Constitution on patent protection was driven by a desire for justice more than a desire for economic benefit.

There may, of course, be valid reasons to see interconnections between political philosophy and economic success; the striking examples of our times were the Communist and Socialist movements (originally, these were actually one single movement), which at their root rejected private ownership as a positive societal value. The facts of the last 100 years are that, in general, even countries with abundant natural resources and well-educated populations did poorly in economic terms under Communist and Socialist governments. But the emphasis of the founders of the U.S. would seem to be on the philosophical and political benefits of private owership (which, in their view, included intellectual property) rather than on the economic benefits thereof.

It is, however, still well within U.S. intellectual tradition to examine and advocate other approaches. Traditional U.S. emphasis on freedom is often translated into an emphasis on freedom of information, which seems in tension with the idea of private ownership and control of intellectual property. An example of such position may be found in the book *Against Intellectual Monopoly* (16) by Michele Boldrin and David K. Levine. The authors' main thesis is that intellectual property is, in fact, an "intellectual monopoly" that does not directly promote innovation, one that it hinders rather than helps free-market competitiveness. From this, they infer that intellectual property protection has a diminishing effect on innovation and wealth. Their main argument seems to revolve around their assertion that "the only justification for intellectual property [protection] is that it would increase — *de facto* and substantially — innovation and creation."

This argument may be attractive on its surface, but, like the economic and legal arguments in support of slavery, can lead via extrapolation to undesirable results. For example, one can use it to dismiss the need for a criminal code based on the observation that laws restrict freedom, and no law can make a law-abiding citizen out of a criminal. The way, in our view, to resolve such apparent tension in reasoning is to distinguish between the concrete benefit and tangible application of freedoms for individuals versus the *abstract* benefit and vague application of the notion of "freedom for society at large." As scientists, we understand the flaws in an argument that can result from such an application of a line of reasoning outside of its domain of validity. We assert that the flaw in the above argument is similar: Arguments about the "freedom for society at large" are platitudes that state a general objective, but declarations of the specific rights of individuals (especially, the rights of individuals vis-a-vis the government) can be the basis for specific legal principles. Comparing benefits of platitudes to those of specific, actionable principles is an inherently invalid comparison.

Of course, some scholars (for example, Boldrin and Levin (16)) go even farther and argue that intellectual property is detrimental to progress in science and, therefore, should be abolished. Our answer to this argument is that society needs not only science but also engineering (e.g., the application of scientific principles to create practical devices and effects) and that on both the basis of justice and economic benefit, engineering and the creation of such practical devices benefit from intellectual property protection.

Some even assert that intellectual property protection hinders not only science but also engineering. For example, Mike Palecek in "Capitalism Versus Science" (17) seems to hold a very similar opinion to Boldrin and Levin's when he writes:"We are constantly bombarded with the myth that capitalism drives innovation, technology, and scientific advancement. But in fact, the precise opposite is true. Capitalism is holding back every aspect of human development, and science and technology is no exception."

There is, as can be seen, tension between the goals of protection of private property and the goals of freedom of information. Of course, many aspects of society may exhibit such tensions; for example, the existence of a criminal code is one of such aspects, as we have indicated above. The hard part is to reach agreement within society regarding what constitutes an appropriate balance of these tensions.

The above discussion shows that the "marketplace of ideas" still has many streams on the subject of the political and philosophical basis for intellectual property protection. But the general American philosophy emphasizes justice to the individual creator as the overriding argument, which led to the inclusion of intellectual property protection in the U.S. Constitution and legal system.

OTHER CONSIDERATIONS

Another question to consider about software patenting is whether intellectual property in general, and computer algorithms in particular, are at present over-protected. Judging from various market indications and economic trends, that does not seem to be the case. While the prices of software and products whose functionality depend on software show a definite tendency to decrease in price, the prices of other items, such as energy, raw materials, and food, are steadily going up. This would seem to indicate that the existence of software patents in their current form are not artificially raising the price of software in the market. This decrease comes despite the fact that software is in general created by highly-trained and usually well-compensated people. Note that in the absence of effective protection of intellectual property, some would argue that such compensation would likely become unsustainable — the period of exclusivity and control offered by software patents creates the opportunity to earn back the up-front expense involved in creating the software, and a major portion of that up-front expense is salaries for the creators of that software.

WHAT'S IN THE FUTURE?

Gregory Clark, in his seminal book *A Farewell to Alms: A Brief Economic History of the World* (13), provides an analysis of the Industrial Revolution in England around 1800. We believe that there might be some similarities between the Industrial Revolution then and the Information Revolution now, and therefore Clark's work might offer some perspective on the software patenting controversy.

Both revolutions were predicated upon unprecedented, large-scale innovation; the driving forces during the Industrial Revolution were the coal and textile industries, and the corresponding driving forces during the Information Revolution are the software/information and micro-electronics industries. Clark demonstrates that, in general, during the Industrial Revolution, it was not always the innovators who were the main beneficiaries "of the social rewards their enterprise wrought." According to Clark, "the textile innovators of the Industrial Revolution, even those who were successful and are now famous, typically earned small returns; wage earners and foreign customers, not entrepreneurs, were the overwhelming beneficiaries of innovation." Similarly, we argue that principal beneficiaries of the Information Revolution have been the members of the general public, who have the ability to make use of amazing products and services at very modest prices; the next big class of beneficiaries are the large number of workers who earn a living providing those products. The actual innovators at times did ***not*** benefit significantly. Returning to our example above, it was ***not*** the inventor of the computer spreadsheet who made the fortune; instead, it was those who re-marketed the (at the time, unpatentable) spreadsheet who made the fortune. The lack of software patents during a seminal period of the Information Revolution seems to have cut out an entire generation of inventors from reaping the benefits of their creations. It is not yet clear if the subsequent permitting of software patenting has brought us to a more balanced status quo or not.

REFERENCES

1. Merriam Webster Dictionary. Springfield (MA): Merriam Webster Inc.; c2018 [accessed 2018 Feb 15]. https://www.merriam-webster.com/dictionary/.
2. Fusco S. Is in re Bilski a déjà vu? Stanford Technol Law Rev. 2009;45(1). [accessed 2018 Feb 15]. http://stlr.stanford.edu/pdf/fusco-bilski-deja-vu.pdf.

3. Syllabus. Alice Corporation Pty. LTD v. CLS Bank International et al. 19 June 2014.
4. Nazer D, Ranieri V. Bad day for bad patents: Supreme Court unanimously strikes down abstract software patents. Electronic Frontier Foundation. 2014 Jun 19 [accessed 2018 Feb 15]. https://www.eff.org/deeplinks/2014/06/bad-day-bad-patents-supreme-court-unanimously-strikes-down-abstract-software.
5. Alice Corp. v. CLS Bank International. Wikipedia, The Free Encyclopedia. [accessed 2017 Dec 1]. https://en.wikipedia.org/wiki/Alice_Corp._v._CLS_Bank_International.
6. Patent Process Overview. Alexandria (VA): USPTO; [2017 Jun 9; 2018 Feb 15]. https://www.uspto.gov/patents-getting-started/patent-process-overview.
7. Orozco D. Administrative patent levers. Penn State Law Rev. 2012;117(1)1-51.
8. Business Method Patents: Hearing before the House Subcommittee on Courts, the Internet, and Intellectual Property, 107th Cong., 1st Sess. (April 3, 2001).
9. Decision of the Federal Court of the Northern District of California. Netflix, Inc. v. Rovi Corp. 2015, which in turn cites from Mackay Radio & Telegraph Co. v. Radio Corp. of America, 306 U.S. 86, 94 (1939).
10. Klemens B. Math you can't use: patents, copyright, and software. Washington (DC): Brookings Institution Press; 2005.
11. Asimov I. Understanding physics. New York (NY): Barnes & Noble; 1966.
12. Arrow K. The Rate and direction of inventive activity: economic and social factors. Princeton (NJ): Princeton University Press; 1966. 609-626.
13. Clark G. A farewell to alms: a brief economic history of the world. Princeton (NJ): Princeton University Press; 2007.
14. Posner R. An economic theory of privacy. Regulation. 1978;19-26.
15. Case studies in emerging software eco-systems. Int J Adv Comp Sci. 2011;1(1).
16. Boldrn M, Levine DK. Against intellectual monopoly. New York (NY): Cambridge University Press; 2008.
17. Palecek M. Capitalism versus science. In defence of Marxism. 2009 [accessed 2018 Feb 15]. https://www.marxist.com/capitalism-versus-science.htm.
18. U.S. Constitution, article I, section 8.