The Washington Post Print

Will the Supreme Court save us from software patents?

By Timothy B. Lee, Updated: February 26 at 1:13 pm

If you write a book or a song, you can get copyright protection for it. If you invent a new pill or a better mousetrap, you can get a patent on it. But for the last two decades, software has had the distinction of being potentially eligible for both copyright and patent protection.

Critics say that's a mistake. They argue that the complex and expensive patent system is a terrible fit for the fast-moving software industry. And they argue that patent protection is unnecessary because software innovators already have copyright protection available.

This spring, the Supreme Court will weigh in on the patentability of software for the first time in a generation. In the 1970s, the high court placed strict rules on software-related patents. But since then, a lower court has effectively overruled the Supreme Court's precedents, allowing hundreds of thousands of legally dubious software patents to be approved.

The arguments in the software patent debate have barely changed since the 1970s, but the players in the debate have changed radically. In 1972, IBM was a leading software patent opponent. Today, Big Blue has become one of the concept's biggest supporters. In 1991, Bill Gates warned that patents could bring the software industry to a "standstill." Today, Microsoft is fighting to protect the tens of thousands of software patents in its portfolio.

Yet today's big companies got it right the first time around: Patents are a poor fit for the software industry. Software patents create a lot of unnecessary litigation while doing little to encourage innovation.

Mental steps

The first high-profile battle over software patents pitted two of America's most powerful companies, AT&T and IBM, against each other. Gary Benson and Arthur Tabbot were researchers at AT&T's legendary Bell Laboratories, and in 1963 they sought a patent on a method of converting between two different binary formats.

Their invention worked like this. Suppose an AT&T customer dials "202." In binary, 2 can be represented as "0010" and 0 as "0000." So as the customer dials "202," AT&T's computer system might store this as "0010 0000 0010." This format is known as "binary coded decimal."

But it's more efficient for a computer to represent 202 in its "pure binary" form: "11001010." Benson and Tabbot had developed a sequence of mathematical operations that allowed computers to rapidly convert a binary-coded decimal number like "0010 0000 0010" into the

more compact form "11001010."

The Patent Office rejected their application, arguing that the pair were merely seeking to patent "a logical list of mental steps." Mathematical formulas had long been off-limits for patent protection. The Patent Office believed that the sequence of mathematical operations described in Benson and Tabbot's application did not become a patentable invention simply because they were carried out by an electric circuit rather than with a pencil and paper.

In 1972, IBM was the world's largest computer company. When the case reached the Supreme Court in 1972, IBM filed a brief opposing the patent — and software patents in general.

Until 1968, IBM had given its software away for free to customers who purchased its hardware. In the company's view, allowing patents on software would "have the inevitable effect" of "stifling developments in computer programming" — thereby limiting the market for IBM's expensive mainframe computers.

IBM also argued that copyright protection was a better fit for the software industry than patents. "The need for protection against copying is very real," IBM conceded. "Copyright achieves the basic purpose of preventing one who has made no investment of time or money in creating a computer program from appropriating the program by copying it."

But crucially, copyright protection allows someone to independently develop software to achieve the "same overall result" as a copyrighted program. In contrast, patent law doesn't allow independent invention limiting the opportunities of future innovators and creating the risk of accidental infringement and wasteful litigation.

The case for software patents

While IBM made the case against software patents, one of IBM's fiercest adversaries was making the case for them: a trade group called the Association of Data Processing Service Organizations (ADAPSO), which represented some of the first software companies.

The very concept of an independent software company was new in 1972. ADAPSO's members were looking for ways to break Big Blue's stranglehold over the software business. They were enthusiastic supporters of the antitrust lawsuit the government filed against IBM in 1969. And some ADAPSO members saw patents as another weapon they could use against what they viewed as a domineering monopolist.

A leading figure in ADAPSO was Marty Goetz, an executive at the software company ADR. ADR produced one of the first commercial software products, called Autoflow. "For five years, we were selling our software and IBM was giving away their software," Goetz says. "AutoFlow ended up having a competitor, a very inferior product by IBM." Goetz says Autoflow was the superior product, but IBM's lower price and promises of future improvements made customers reluctant to become Autoflow customers. "That was one reason why we got a patent," Goetz said.

At Goetz's urging, ADAPSO filed a brief in favor of software patents. ADAPSO made arguments that are still being made by software patent advocates today.

ADAPSO worried that if its members produced a successful software product, "IBM and the other hardware manufacturers will be able to copy the technological concepts of the priced software and put out their own version" for free. In ADAPSO's view, patents were needed to protect software companies from this "devastating prospect."

ADAPSO also objected to the government's (and IBM's) argument that software patents were patents on mental steps. "Software is actually a form of machine device which structures the general-purpose hardware," the trade group wrote. In other words, when you load a computer program into a computer, you effectively turn it into a new type of machine that's eligible for patent protection.

A skeptical court

The Supreme Court sided with the government and IBM. "One may not patent an idea," Justice William Douglas wrote for the court. "But in practical effect that would be the result if the formula for converting BCD numerals to pure binary numerals were patented in this case." The BCD-to-decimal formula has "no substantial practical application except in connection with a digital computer," so Benson and Tabbot's patent "in practical effect would be a patent on the algorithm itself."

The Supreme Court doubled down on this skeptical stance six years later. An inventor named Dale Flook applied for a patent covering a new formula for setting an "alarm limit" during a catalytic conversion process. An alarm limit is a number that signals that a variable such as temperature or pressure has reached an unsafe level.

The Supreme Court had rejected the Benson patent because it claimed all conceivable applications of the BCD-to-binary conversion formula. Flook argued that his own patent was different, for two reasons. First, it was limited to a specific industrial application: the catalytic conversion of hydrocarbons. And second, unlike in the Benson case, the computer in the Flook process does something tangible with the result of the calculation: modifying the limit that triggers safety alarms.

But the Supreme Court didn't buy it. The fact that Flook included the obvious step of updating the alarm limit in his patent claim didn't change the fact that it was fundamentally a patent on a mathematical formula, Justice John Paul Stevens ruled for the court. "The notion that post-solution activity, no matter how conventional or obvious in itself, can transform an unpatentable principle into a patentable process exalts form over substance," Stevens wrote. Flook was effectively seeking ownership of a mathematical algorithm. The Supreme Court said no.

Software patents make a comeback

As the 1980s dawned, things looked grim for advocates of software patents. Patent examiners were routinely rejecting applications for patents on computer-related inventions. But in 1981, the Supreme Court finally found a computer-related patent it liked.

To produce rubber products, raw rubber is placed into a rubber molding press and heated up. Good results depend on opening the press at precisely the right time. Inventors Thomas Diehr and Theodore Lutton realized that they could improve the rubber-curing process if they constantly measured the temperature inside the press, fed those measurements into a

computer, and used the data to recalculate the best time to open the press.

The Patent Office wanted to reject the patent based on the Supreme Court's previous software patent decisions. But a narrow majority of five Supreme Court justices disagreed. Justice William Rehnquist wrote that Diehr and Lutton "do not seek to patent a mathematical formula. Instead, they seek patent protection for a process of curing synthetic rubber" that happens to employ a mathematical algorithm as one of its steps.

The decision complicated the job of the Patent Office. The Diehr decision did not overrule the Flook and Benson decisions; the kind of "pure" software patent rejected in those earlier decisions was still illegal. But inventions that tied a computer program to a real-world application, like Diehr and Lutton's rubber-curing technique, became permissible. And no one was sure where to draw the line between the two.

"We had this anarchy within the USPTO about how these decisions would be interpreted in the office," says Bruce Lehman, who became head of the patent office in 1993. In the decade after the Diehr decision, there were few court decisions clarifying which computer-related inventions were eligible for patent protection and which were not. Different patent examiners "had different views on things and they were coming out in different places," according to Lehman.

As the Patent Office allowed more software patents, many in the software industry reacted with alarm. In the early 1970s, some software companies wanted patents to use as a weapon against IBM. But a new generation of software companies had thrived without patent protection, and they worried that software patents would just bring unnecessary litigation.

"If people had understood how patents would be granted when most of today's ideas were invented, and had taken out patents, the industry would be at a complete standstill today," wrote Microsoft CEO Bill Gates in a 1991 memo to his executives. Microsoft was still a relatively small company in 1991, and Gates worried that "some large company will patent some obvious thing," which could give the company "a 17-year right to take as much of our profits as they want."

Gates took a pragmatic approach; the company got to work filing hundreds, and eventually thousands, of patent applications. But other software companies began speaking out publicly.

In 1994, Lehman organized public hearings on software patents and invited leading software companies to submit comments. The database company Oracle argued that patent protection was "not appropriate for industries such as software development in which innovations occur rapidly, can be made without a substantial capital investment, and tend to be creative combinations of previously-known techniques." Oracle's views were echoed by Adobe, creator of popular applications such as Photoshop.

IBM had opposed software patents in the 1970s, but by the 1990s Big Blue had changed its mind. "We can't divorce computer program-related inventions from computer hardware and other microprocessor inventions," IBM argued, echoing an argument its adversaries had made two decades earlier. "As the [computer] industry matures and competition from overseas increases, patents will be the key to protecting the most valuable US-originated innovations," Big Blue argued. Microsoft also endorsed software patents despite the private reservations of its CEO.

The new sheriff

During the 1970s, patent law was shaped by a Supreme Court that was skeptical of patents on software. Even its 1981 decision emphasized that there were limits on software-related patents. In 1982, Congress made a seemingly innocuous change to the structure of the court system that had a profound impact on the legal status of software patents.

Most areas of the law are handled by generalist judges organized into a dozen geographically based appellate courts. But Congress, concerned that patent law had become too complex for generalist judges, created a new court called the Federal Circuit Appeals Court. The Federal Circuit was given jurisdiction over all patent appeals. And perhaps because its judges spend so much time rubbing elbows with patent attorneys, the new court would prove to have a strong pro-patent bias.

An important turning point came in a 1994 ruling involving a computer graphics technique called anti-aliasing. When a computer draws geometric shapes in black and white, they can have a jagged, blocky appearance. Anti-aliasing is a technique to give shapes a smoother appearance by drawing their edges in shades of grey.

Anti-aliasing algorithms had been well-known in the computer science community since the 1970s. That didn't stop a team led by Kuriappan Alappat from applying for a patent on the concept of using anti-aliasing to improve the display of a laboratory device known as an oscilloscope.

The patent application didn't include details about the oscilloscope's hardware; indeed, its claims were broad enough to cover a standard PC running oscilloscope software. The government argued that Alappat was seeking to preempt the anti-aliasing algorithm just as AT&T had tried to preempt the BCD-to-decimal algorithm two decades earlier.

The Federal Circuit disagreed. "A general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software," the court held. This is, of course, exactly the argument that software patent advocates like ADAPSO had made in the 1970s. Those arguments had not swayed the Supreme Court. But the Federal Circuit was more sympathetic.

The patent court pushed the envelope even further in 1998. State Street Bank challenged a patent on the use of a computer to implement a strategy for managing mutual funds. This patent had even less connection to a specific machine than the Alappat patent did. But the patent appeals court approved it anyway.

"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," the court ruled.

If the Alappat ruling pushed the Supreme Court's precedents to the limit, the State Street decision seemed to ignore them altogether. Pam Samuelson, a scholar at the University of California Law School, says it's "not possible" to square the State Street ruling with the Supreme Court's earlier precedents. "They didn't like the ruling and so they gave it a narrow interpretation," she argues. "In effect, they overruled it. They thought the Supreme Court was

going to continue to pay no attention, and so they had a free hand."

Unsurprisingly, the number of software patents soared during the 1990s. And the number of lawsuits involving software patents began to soar as well:

A landmark 2008 book found that software patents are more than twice as likely as other patent categories to become involved in litigation. And "business method" patents like the one the Federal Circuit approved in the State Street case were six times as likely to lead to patent lawsuits.

The Supreme Court pushes back

At first, the Supreme Court let the Federal Circuit shape patent law with minimal interference. But by the time John Roberts became chief justice in 2005, the Federal Circuit's patent-friendly jurisprudence had produced record levels of patent litigation and an avalanche of bad press. So the Supreme Court began to give the Federal Circuit closer scrutiny.

The high court didn't like what it found. The first few patent rulings of the Roberts era were unanimous or near-unanimous reversals of the lower court's decisions.

Roberts became exasperated by the Federal Circuit's casual attitude toward Supreme Court precedents. During a 2009 oral argument, he commented that lower courts "don't have a choice" about following Supreme Court precedents. "They can't say, I don't like the Supreme Court rule so I'm not going to apply it." Then, he added wryly: "other than the Federal Circuit."

Of course, the Federal Circuit *is* supposed to follow Supreme Court's rules. But it hasn't done a very good job. In 2012, the high court unanimously overruled a Federal Circuit decision allowing patents on medical diagnostic techniques. Last year, the high court rejected patents on human genes, which the Federal Circuit had previously approved.

Alice in patentland

On March 31, the Supreme Court will finally return to the software patent issue after a 33-year hiatus. The case focuses on a patent that claims the use of a computer to eliminate "settlement risk," the risk that one party will renege on a deal, leaving the other party holding the bag. The patent is owned by a financial institution called the Alice Corporation. Like the mutual fund patent the Federal Circuit upheld in 1998, the process described by Alice's patent takes place entirely inside the computer, and isn't tied to any specific machine or industrial process.

Alice argued that the fact that the patent required the use of a computer was sufficient to make it a patentable invention. When the case reached the Federal Circuit, the court agreed in a 2 to 1 vote.

But the dissenting judge, Sharon Prost, was furious. "The majority has failed to follow the Supreme Court's instructions—not just in its holding, but more importantly in its approach," she wrote. "Just a few months ago, the Supreme Court reversed us for a second time in its last three terms, hinting (not so tacitly) that our subject matter patentability test is not sufficiently exacting."

Her dissent prompted a larger panel of Federal Circuit judges to take up the case. The result was a mess, as no single opinion gained majority support. Five judges wanted to invalidate the patent outright. But the other five judges wanted to uphold at least some of the claims.

In a dissenting opinion signed by three of her colleagues, Judge Kimberly Moore worried that rules against patenting mathematical algorithms and other abstract ideas were "causing a free fall in the patent system. This case is the death of hundreds of thousands of patents, including all business method, financial system, and software patents as well as many computer implemented and telecommunications patents."

Of course, critics would say that's the point: the Supreme Court placed limits on software patents in the 1970s, and the Federal Circuit is supposed to be enforcing them. In their view, business method and software patents shouldn't have been granted in the first place.

But Moore's comments highlight the dilemma the Alice case creates for the Supreme Court: rightly or wrongly, the Patent Office has issued hundreds of thousands of software patents since 1998. Those patents are worth billions of dollars to companies like IBM, Microsoft, and Oracle. The justices will be reluctant to write an opinion that could decimate the patent arsenals of some of America's wealthiest companies.

But Richard Stern, the Justice Department lawyer who successfully urged the Supreme Court to limit software patents in the 1970s, argues that these considerations shouldn't deter the Supreme Court from upholding the law. "When you make your business model on the basis of what is clearly a wrong interpretation of the law, you don't get a vested interest," he says.

And while a decision invalidating thousands of software patents could be bad for the bottom lines of some large software companies, it could be good for the industry as a whole. One influential 2012 study found that litigation by "patent trolls" cost defendants at least $29 billion in 2011. Those costs are disproportionately due to software patents. A ruling against software patents might destroy billions of dollars worth of patents, but it could save hundreds of billions of dollars in litigation costs.

The arguments about software patents in the 1970s and 1990s were theoretical. Opponents predicted that software patents would have a detrimental impact on software innovation. But the concept was so new that there was little hard evidence on the question.

But now the evidence is in. And it vindicates the skeptical stance taken by IBM in the 1970s and by Gates, Oracle, and Adobe in the 1990s. Today, IBM, Oracle, and Adobe all have thousands of software patents, and unsurprisingly they're no longer opposed to software patents. But software patents have had all the negative effects they predicted it would. Innovative companies have been forced to divert resource from hiring engineers to hiring patent lawyers. Large companies that are no longer on the cutting edge have been using broad patents to demand cash from smaller, more innovative companies. Many programmers and software entrepreneurs view patents as more a nuisance than a reward for innovation.

The Supreme Court has the chance to fix the problem by making clear that it was serious about the limits it established in the 1970s. As the justices weigh their options, it should not only read IBM's 2014 brief making the case for software patents. They should also read IBM's 1972 brief opposing them.