The Software Protection Debate

Tri Ngo
Richard Sinn
December 19, 2005
6.901 Final Paper
Professor Robert Rines

Abstract
Software patents have been a controversial topic for a very long time. The unique characteristics that govern software distinguish it from any other product that has been historically patented. Software has existed and flourished before they were largely patented, and it should be continued to do so without patents. Patents are not needed to encourage innovation when it comes to software; innovation is an inherent characteristic of software. Patents can actually discourage the independent inventor from innovating because of fear of expensive litigation resulting from unknowingly infringing on a patent.

Introduction

Although patents were originally designed to protect the inventor and promote innovation, we believe that patents with respect to software do not. In fact, software patents sometimes impede innovation, and can actually be harmful to the independent inventor.

The United States Constitution declares that the purpose of patents and copyrights is to "promote the progress of science and the useful arts by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries." In other words, a patent protects an inventor from imitators, and gives the inventor incentive and financial compensation for the cost of his or her pains. If patent protection did not exist, a potential inventor, especially an independent inventor, might decide against inventing because of the expenses and risks associated with inventing and putting the invention into form.

However, the idea of software patents has been historically controversial. Many prominent figures in the "software world" oppose the ideas of software patents. There are also many international movements against the idea of software patents, and a few foreign governments have taken steps to restrict software patents. Even in the United States, software patents were difficult to obtain until only recently.

People often confuse software patents with software copyrights, but there is much difference between the two. Software programs have always been copyrightable, but patenting software has not always been straightforward. Because software programs are written, they automatically fall under copyright protection laws, but there has been much debate as to whether or not the process and algorithm behind software programs can be patented. Copyrighting software gives protection to the expression of the written program, but not to the actual implementation or idea behind the code. Thus getting patent protection on a software program would give the inventor stronger protective rights.

Controversy

Patents are generally considered beneficial in most industries, but patents in the software industry are surrounded by controversy. Software tycoon Bill Gates in a memo wrote,

> "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. ... The solution is patenting as much as we can. A future startup with no patents of its own will be forced to pay whatever price the giants choose to impose. That price might be high. Established companies have an interest in excluding future competitors." (Lessig 2002-07-24: Keynote to OSCON)

On the other ideological extreme is open source software advocate Linus Torvalds who humorously writes,

> "The fact is, technical people are better off not looking at patents. If you don't know what they cover and where they are, you won't be knowingly infringing on them. If somebody sues you, you change the algorithm or you just hire a hit-man to whack the stupid git." (http://lwn.net/Articles/7001/)

These statements from representatives of the software industry demonstrate the dubious nature of patents in this industry.

In fact, some countries have attempted to move beyond the controversy by attempting to eliminate software patents. Article 52 of the European Patent Convention (EPC) excludes "*schemes, rules and methods for performing mental acts, playing games or doing business, and* ***programs for computers"*** from patentability. However, diverse interpretations of this and related statements in the EPC have allowed many software patents to be granted despite this law. In response, certain lobbying groups throughout Europe and the rest of the world have launched a concerted effort to ensure the elimination of software patents.

The appearance of recent movements from within the software industry has also cast a shadow of doubt on the appropriateness of software patents. The open source software movement and the industry formed around it have felt threatened by the existence of software patents. Open source developers and companies argue that software

patents impede their ability to create new software due to fear of litigation from the unknowing infringement of an existing software patent.

Legally, software patents exist on slippery foundations. Unlike physical inventions, abstract software code is difficult to categorize. Some see software as a mathematical algorithm and software innovation as the discovery of a new algorithm. Mathematical algorithms can been seen as a product of nature and as such are ineligible for patent protection.

In response to this controversy, economists have attempted to provide real world evidence of the effects of patents in the software industry. Ironically, these studies have increased the controversy rather than quelling it. Due to the expansive nature of economic studies, researchers have found it difficult to fully eliminate confounding variables. Unlike the physical sciences where hypotheses can often be tested directly through a carefully designed experiment, the creation of an economic experiment to test the effects of patents in the vast software industry is difficult. Researcher Robert W. Hahn concludes in his article, "An Overview of the Economics of Intellectual Property Protection," that "...despite its long history, the literature on intellectual property rights has found few hard conclusions." (37)

Landmark Cases

The United States Patent and Trademark Office (USPTO) website says that under the patent law statute, only “process, machine, manufacture, or composition of matter” can be patented, while mathematical algorithms or scientific truths cannot be patented.

Originally, the USPTO considered software to be a mathematical algorithm, and thus not patentable. However, through the years, the ability to patent software has grown significantly easier.

In 1981, the United States Supreme Court resided over the landmark case *Diamond v. Diehr* (www.law.uconn.edu). The Supreme Court decided that a machine controlled by a computer program was patentable. Although it did not fully allow a computer program to be patentable by itself, "it set the stage for later precedent-setting decisions that extended protection to software." (Hahn 2). The ruling meant that even if most of the invention under review was composed of a computer program or a mathematical formula, it could be patentable as long as the invention as a whole met the requirements of a patent. Through this ruling, "creative patent attorneys were now able to wrap software innovations into patents for tangible processes or products." (Hahn 2).

The 1998 Supreme Court case, *State Street Bank & Trust Company v. Signature Financial Group, Inc.* further eased software to be patentable. In 1999, only one year after the "State Street Decision," John W. Rees wrote that the decision immediately triggered a "boom in [software] patent application filings." Although the Supreme Court decision was originally directed toward business method-related software, it eased the restrictions on the patentability of software in general. The State Street decision basically stated that anything could be patented as long as it could "produce a useful, concrete and tangible result" (Arnold 2001). This "boom" in software patents still exists, and experts estimate that there are now thousands of software patents, with another 10,000 patent pending (Syrowik 2003).

Ironically, as patent protection for software increased throughout the years, copyright protection has weakened. Early on, court rulings tended to favor software copyright holders, but copyright protection was substantially weakened in the case of *Lotus Development Corp. v. Borland International, Inc.* (Hahn 3). In 1996, the Supreme Court upheld without comment the decision made by the First Circuit. This was the case that essentially established the idea that copyright protection for a software program could only be extended to the expression of the program, and that anybody could duplicate the code "provided one does not copy the literal code" (Lundberg 55).

Characteristics of Software Innovation and the Software Industry

The controversy that surrounds software patents may have its source in the unique characteristics of the software industry and software itself. Software is fundamentally different from any other invention or innovation the patent system ever had to deal with. Software is not limited to any physical constraints, and as a result, it has created "a different kind of industry with its own particular economic structure" (Irlam).

Gordon Irlam and Dr. Ross Williams argue that because software is free from physical constraints, it's much more complex than any other industry. Software's complexity has grown so much that some computer programs cannot be understood by a single person. A typical industry can have a product that is comprised of twenty parts, and a more sophisticated industry such as consumer electronics can have products with a thousand parts. However, a software program can comprise of millions of lines of codes, and an uncountable number of parts.

Another aspect of software that makes it more complex is the abstraction techniques of computer programming. Software programs are abstracted into components to be used in larger software programs, and in turn abstracted again to be used in even larger programs. Therefore, "software's abstraction makes it difficult to partition these technologies" (Irlam). Irlam and Dr. Williams continue to argue that most industries have products that are covered by only a few patents, but in the software industry a "product can contain thousands of inventions, any of which might be patented." Because of the ability for one software program to contain many potential patents, and because of the difficulty to partition the many innovations within one software program, it is difficult to analyze each software program.

To add to the complexity of software, the software industry is very dynamic. It develops much faster than any other industry, and new products in software come out every few months, while other more typical industries "typically produce a new generation of products every ten to twenty years" (Irlam). Therefore, during the twenty year term of a patent, many generations of software programs can come and go. A very well known example of this fast evolution in software can be seen with the Windows operating system by Microsoft. Microsoft released Windows 3.1 in 1992, and only nine years later Microsoft released Windows XP, the most used operating system in the world today. In between those nine years, Microsoft went through many significant changes and innovations to their product. To compare the speed of innovation in the software industry with some other conventional industry let's take a look at the media storage industry with respect to the VHS and DVD. The VHS format was released in 1976, and a

new widely accepted standard in video and media storage did not appear until the DVD came out in 1996, almost 30 years later.

The software industry is also very different economically from many of the other industries out there. The aircraft industry has medium research costs, high development costs, and high production costs. Conversely, Irlam and Dr. Williams state that the software industry has low research costs, high development costs, and low production costs. Irlam and Dr. Williams argue that software has such low research costs because "development has not been able to keep up with research" and that software has a high development cost because "it takes a lot of human effort to write production-quality software." Production costs for software is low because it does not cost much money to copy code.

Originally, patents were intended to encourage inventors to invent by incurring the costs of innovation by promising exclusive rights on the new invention for a limited amount of time. However, in an industry where the cost to research and produce may be minimal anyway, it is unclear if software patents serve this purpose. Software patents may actually be devastating to small companies because litigation over patent infringements can be very expensive to both parties, and it could destroy small companies.

Because of these unique characteristics in software innovation, the software industry innovates in a sequential and complementary way. Bessen writes "The sequential and complementary nature of innovation is widely recognized, especially in high-tech industries" (3). The innovation in the software industry is sequential because every new invention builds directly upon a previous one. Additionally, since innovations

in the industry often occur through the complementary efforts of many inventors, software innovation is labeled as being complementary. In light of these characteristics, patents can actually impede innovation. Patents can interrupt the sequential innovation mechanism by preventing the use of a prior invention as a basis for further innovation. For example, patents can restrict competitors from contributing ideas that may actually help achieve new innovations. In an industry that exhibits sequential and complementary innovation, a firm that patents its product "can prevent its competitors from using that product to develop further innovation" (Bessen 3).

The Open Source Movement

The recent open source software movement is opposed to software patents, and refuses to take out patents on any of their new innovations, and yet it has been surprisingly successful.

The open source movement can be interpreted as a response to the tension in the software industry regarding intellectual property rights. Unsatisfied with the proprietary development model, open source software wishes to make explicit the implicit characteristics of the software industry. The movement aims to guarantee the right of the software innovator to use prior technology as a basis for innovation. The movement also encourages extensive collaboration on the development of software. These two aims run parallel with the unique industry characteristics of sequential and complementary innovation.

Many reliable and praiseworthy software programs have been created as result of the open source movement. Many academics and software scholars are avid supporters of the open source movement. Tim Berners-Lee, Richard Stallman, and Linus Torvalds are a few names that are commonly associated with open source software. They and many others believe that because software was borne out of academic spirit, information should be free and available to everybody.

Commonly Suggested Solutions

Although there is much debate as to whether patents benefit or harm the software industry, it is the consensus that tensions and problems exists concerning software patents. In order to alleviate these problems scholars have suggested some improvements to the current treatment of software patents. These suggestions all attempt to place patent policy in agreement with the sequential and complementary innovation present in the software industry. To improve the current system, some have suggested that the obviousness barrier to obtaining a software patent be heightened, making it more difficult for the USPTO to grant a software patent. Other suggestions include that the scope of the granted patents should be narrowed, which would make the number of patents actually granted much lower. The biggest problem with all these suggestions is that it is difficult to quantify these regulations, which is why it was a problem in the first place. It is very difficult, if not impossible to sometimes quantify what "obviousness" means, or what "novelty" means.

Another idea that is commonly suggested is that software patents should be banned all together. Although this may sound like an extreme proposition, the basis for

this argument is well thought out and convincing. There are a lot people who support this idea, and many of them are accredited computer scientists and software programmers.

Our Proposed Solution and Reasoning

In order for the software industry to develop efficiently and software to innovate to its potential, it would be best to ban software patents. Software patents not only hinder the progress of software innovation, but it has grown to the point where software patents are almost useless.

Patents do not make sense for the software industry in many ways. First of all, the software world is so dynamic that taking a twenty year monopoly out on a particular software invention does not make much sense. During those twenty years, several different generations of a product can be innovated and produced, and a software invention that was non-obvious twenty years ago may seem very obvious only two years later.

Moreover, each software program can potentially contain hundreds if not thousands of different innovations and inventions that can qualify as a patent under current law. This makes developing new programs very difficult because programmers and developers have to make sure that their new software programs, which can also contain hundreds or thousands of new innovations, do not violate any existing patent. All this could be very taxing on small companies and independent inventors because they might not have the resources to search through thousands of patents to make sure that their innovations in their software program do not violate any existing patents. This

would not only slow down innovation, but also discourage small companies or independent innovators from innovating for fear of litigation.

In regards to software, the industry will be able to thrive even if patents did not exist. The cost to produce software is so low, that a form of compensation from patents is not needed. Also, because of the unique characteristics of the software industry, innovation is a necessary quality a company or an inventor must have in order to be successful in the software world. Thus, the software world does not need patents to encourage people to innovate; it is already inherent in the industry.

Software patents have grown to the point where major software companies are in a patents "arms race." Big software companies try to take out as many software patents as possible to protect themselves from other software companies that also have many software patents. The reason for this behavior is because if one big company decides to sue another big company over patent infringements, the other company will be able to counter-sue with their own claims of patent infringement. The idea is that because there are so many patents out there, the chances of someone infringing on a software patent is very high, and that each company must be infringing on a patent somewhere. Therefore, the more patents a company owns, the more likely it can defend itself if someone decides to sue them. Additionally, if a big software company wants to eliminate a small software company as a competitor, it can sue them for infringing on one of their patents. Fighting litigation in court is very costly, and it can destroy a small software company or an independent inventor whether they win or lose.

Patents in software do not serve the purpose of what a patent should be. It creates economic inefficiency because the industry behaves differently from other more

traditional industries. Patents tend to be more beneficial to the bigger companies, and they are not necessary to stimulate innovation in the software industry. In the early histories of software, software was developed under a patent-less environment that followed from the academic spirit of information sharing. Software was able to thrive in the beginning without any patents, and it should continue to thrive today without patents.

Conclusion

The idea of patents in the United States existed since the drafting of the Constitution, and the United States patent system has existed for two hundred years. Then why is it starting to break down now for software? Software is so fundamentally different from anything we have seen before, that the patent system just does not work for it. The idea that patents are supposed to protect the inventor and promote innovation does not apply to software; software can continue to prosper without patents. At best, the patent system is ineffective for software, and at worst it can be injurious to the entire software industry.

Works Cited

Arnold, Beth and David Lane. "Patent Strategies for Protecting Bioinformatic Inventions: It May be Worth Venturing Out of Group 1600." Foley Hoag, LLP. 1 February 2005. 10 December 2005 <http://www.fhe.com/publications.asp?pubID=000323292105#_ftnref9>

Bessen, James and Eric Maskin. "Sequential Innovation, Patents, and Imitation." researchinnovation.org January 2000. 10 December 2005 <http://www.researchoninnovation.org/patent.pdf>

Hahn, Robert W. Intellectual Property Rights in Frontier Industries. Washington, D.C.: AEI-Brookings Joint Center for Regulatory Studies, 2005.

Irlam, Gordon and Ross Williams. "Software Patents: an Industry at Risk." 25 January 1994. 10 December 2005 <http://lpf.ai.mit.edu/Patents/industry-at-risk.html>

Lundberg, Steven W. and Stephen C. Durant, ed. Electronic and Software Patents. Washington, D.C.: The Bureau of National Affairs, Inc., 2000.

Rees, John W. "'State Street' Decision Causes 'Boom' in Software Patent Filings." FindLaw.com. 1 March 1998. 10 December 2005 <http://library.findlaw.com/1999/Mar/1/128488.html>

Syrowik, David R. and Roland J. Cole. "The Challenge of Software-Related Patents: A Primer on Software-Related Patents and the Software Patent Institute." SPI.org. 9 December 2003. 10 December 2005 <http://www.spi.org/primintr.htm>

The USPTO – General Information Concerning Patents. 2005. USPTO. 10 December 2005 <http://www.uspto.gov/main/patents.htm>

Diamond, Commissioner of Patents and Trademarks v. Diehr. University of Connecticut. 10 December 2005 <http://www.law.uconn.edu/homes/swilf/ip/cases/diamond_di.htm>