Software Patents Considered Harmful

Adrian Stere

March 24, 2006

Abstract

In the mid-nineties, the US Patent Office has begun awarding patents for computer algorithms, in the hope that it would encourage innovation and boost research spending in the software industry. Time has proven this hope baseless. Today, not only are software patents numerically uncorrelated with either innovation or research spending, but they have raised the barrier to entry in the software market, have reduced the level of competition, have vastly increased spending on litigation due to abuses, and have had a chilling effect on open source/free software development and adoption through a host of high-profile lawsuits.

1 Introduction

1.1 History

The practice of awarding patents began in medieval Europe as a means for monarchs to raise funds or reward allegiance by granting selective monopolies on trade in certain goods. Several centuries later, some countries saw the need to encourage scientific and technological advance, and the practice of awarding patents on inventions was one of the tactics employed toward that goal. The use of patents as an incentive for innovation was first recorded in Britain in 1624, when the Statute of Monopolies limited monopoly granting to inventors [7].

The patent system in the United States was designed \textit{ab initio} as a way to reward innovation. The first article of the United States constitution instructed Congress 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”. Thus the three branches of intellectual property law were established: copyright, for abstract creations such as works of art and literature, patents, for inventions with tangible manifestations, and trademarks, for brand identities.

This more or less worked until the second half of the last century, when computers came along. When the question arose in the 1960s as to whether software was patentable, the US Patent Office answered with a resounding “no”, as software was then seen by legislators as being just encoded expressions of mathematical algorithms, which were outside the scope of patents at the time. Throughout the 70s and 80s this attitude has seen progressive relaxation; patents were awarded for software
components of material, tangible inventions at first, then for increasingly abstract inventions, until, in 1994, the decision of the US Supreme Court *In re Alapat* opened the way to granting patents exclusively on computer programs [1].

Since then, software patenting has exploded. Today, approximately 20,000 software patents are granted yearly in the United States, or 15% of all patents [1].

### 1.2 Definition

Under current US legislation, a patent is “a grant formalized by an official document issued by the US Government” [11] having the attributes of personal property. All patent applications must possess the following main attributes: novelty, utility, non-obviousness, and proper disclosure. In other words, to be awarded a patent, an invention must be new and useful, must not be obvious to someone skilled in the trade, its description must be detailed enough for a person skilled in the trade to implement it, and must also lay out a reference implementation.

Once granted, a patent gives the holder full ownership of the invention, including the right to prevent others from using it. Anyone who wishes to use the invention in any way must obtain a license from the patent holder. Implementing the invention without acquiring a license, even when the implementer can prove they were not aware of the existence of the patent, constitutes patent infringement – in contrast with copyright, where proven ignorance is a valid excuse for copyright infringement.

A software patent is “a patent that claims as all or substantially all of its invention some feature, function or process embodied in a computer program that is executed on a computer”.

### 1.3 Why Patent Software

While originally the modern patent system was enacted ostensibly to encourage innovation, today, the philosophy behind patents in general, and software patents in particular, is focused on encouraging the investment necessary to produce and exploit inventions [3]. If companies are granted limited monopolies on technologies they discover, the reasoning goes, their profitability will increase, and they will spend more money on research and development, thus advancing the state of the art and ultimately benefiting the consumer.

### 1.4 The Software Patent Debate

Software-related patents are currently the subject of intense debate worldwide. Some countries, such as the USA and Japan, allow patents to be issued on computer programs. Others, like the European Union, are in the process of deciding whether to allow such patents or not. The main proponents of software patents are lawyers, intellectual property companies, and most software publishers. The main opponents are economists, academics, the open source and free software communities, and some software publishers.
2 The Case Against Software Patents

Software patents are opposed on both philosophical and pragmatic grounds. The philosophical argument holds that the free flow of information regarding computer algorithms, much like mathematical theorems and formulae, is essential to mankind’s intellectual progress, and hence granting ownership over them is morally wrong. The pragmatic argument relies on evidence pointing towards the conclusion that software patents tend to reduce competition, and discourage spending in research and development.

2.1 The Philosophical Argument

The philosophical argument against software patents is especially popular in the academic and free software communities, with Donald Knuth and Richard Stallman being two of its most prominent champions. They both see computer algorithms as something that cannot and should not be owned, lest progress in information sciences and technology be severely deterred.

Stallman argues that with software patenting in effect, one can no longer safely write and distribute software, since it is in effect impossible to find out beforehand what patents may cover one’s design, as “some of the patent applications that are pending are secret” [10]. Hence the very real danger of releasing a computer program, only to find out that it is infringing someone’s patent months, or even years after release.

In a letter to the US Patent Office, Knuth points out that had software patents been in effect since 1980, much of the software we now consider essential to scientific progress could not have been published due to prohibitive licensing fees and the very real fear of getting sued for infringement. This includes Knuth’s own TeX package, “now used to produce more than 90% of all books and journals in mathematics and physics”. [8]

2.2 The Practical Argument

Many software publishers and economists agree that the practice of patenting of computer code has a detrimental effect on innovation, research spending, and especially competition in the information technology world. This argument is based on the belief that licensing fees unfairly raise the barrier to entry in the marketplace, that current US patent law allows patents that are too broad in scope, and thus liable to be used by their holders to block other companies from developing competing products. They also worry that abuses of patent law are leading to excessive litigation and thus have a negative effect on the industry as a whole.

2.2.1 Software Patents Fail to Achieve Their Purpose

Perhaps the strongest argument against pure software patents is that they simply fail to achieve their stated goals. Like many other public policy issues, patents reflect
a social cost-benefit trade-off. In this case, there are both calculated costs, such as having essentially no competition at the forefront of emerging technologies for a limited period of time, and unforeseen costs, given by the unavoidable exploitation of loopholes in the patent legislation. The patent system rests on the assumption that the benefits of having public disclosure of design details of new inventions, and encouraging investment in research and development outweigh the costs.

Software patents violate the first half of that assumption outright, since disclosure of source code, or any kind of implementation details is not required in applying for a software patent. The validity of the second half of the assumption is somewhat harder to measure. However, there are studies that show no correlation between number of patents and R&D spending. The number of patents does, however, correlate with propensity for strategic patenting (i.e., applying for patents for bargaining purposes or to block competitors from developing strategic technologies) [1] Economic models developed by Bessen and Maskin support this correlation; they indicate that “in such a dynamic industry, patent protection may reduce overall innovation and social welfare.” [2]

2.3 Software Patents Discourage Innovation

In the United States, a patent is granted for a computer program based on an application that describes the functionality of the software, but does not mention implementation details. This goes against the philosophy of openness at the foundation of the patent system, making it harder for inventors to draw on each other’s work. It also makes it possible to patent an idea for a system for which no implementation exists, or is impossible to implement with current technology.

The latter, in turn, acts as a disincentive not only for innovation, but also for competition. Being able to patent inventions that do not (or even cannot) exist greatly facilitates the practice of strategic patenting, whereby a company patents more, simply to block competitors. In the absence of a cost deterrent, this practice increases the barrier to entry in the industry, increases litigation, and thus reduces competition.

2.3.1 Software Patents Enable Legal Extortion

Much like mathematical theorems, computer algorithms are highly abstract and general in nature. A patent claim on a computer algorithm could apply to a great number of different technologies, which makes it extremely difficult to develop new technologies without infringing existing patents. Combined with the high cost of litigation, this leads to a situation where patent holders can effectively extort money from other technology developers under threat of lawsuit.

A particularly poignant anecdote that illustrates this point comes from software lawyer Gary Reback. In a 2002 article published in Forbes magazine, Reback recalls his experience defending Sun Microsystems from IBM’s claim that a particular application developed by Sun infringed on seven of IBM’s patents covering an algorithm to transform a line segment into a rectangle. To Reback’s claim that the algorithm
was obvious enough that the patent would not hold in court, IBM’s lawyers had the following to say [9]:

OK, maybe you don’t infringe these seven patents. But we have 10,000 U.S. patents. Do you really want us to go back to Armonk [IBM headquarters in New York] and find seven patents you do infringe? Or do you want to make this easy and just pay us $20 million?

2.3.2 Intellectual Property Companies and Submarine Patents

Other abuses of US patent law include the recent proliferation of intellectual property companies – companies whose sole activity consists of registering patents and collecting licensing fees from anyone who wishes to develop actual products based on technologies covered by their patents. The existence of such companies surely runs contrary to the spirit of patent law; they do not themselves put their innovations to any use, and they impose a cost penalty to anyone who does.

The predatory practice of collecting “submarine patents” has also been adopted by some software publishers as a means to boost revenue. Submarine patenting is the practice whereby a company quietly applies for patents on an algorithm that is about to become an industry standard, sleeps on it until it becomes widespread, then uses it to extort money out of successful implementers. Unisys’ patent on LZW compression used in GIF images and Lemelson’s patents on bar codes are particularly egregious examples of this harmful practice.

A related practice is that of “patent trolling” – obtaining specious, overbroad software patents with the sole purpose of extorting settlements out of implementers under threat of litigation. A few of the more well-known recent examples are Eolas’ patents on web browser plugins, NTP, Inc’s patents on wireless email, Scientigo’s patents on XML technology, and Amazon’s patents on one-click shopping.

2.3.3 Effects on Open Source and Free Software

The recent commercial success and increased adoption rate of open source and free software (FOSS) has also attracted the attention of unscrupulous patent law abusers. The best feature of FOSS – availability of source code – is also its greatest vulnerability in the face of a patent infringement lawsuit. As the recent case of SCO vs. IBM has demonstrated, availability of source code makes it considerably easy for a patent holder to point at a piece of source code (e.g. the Linux kernel) they claim infringes on their patents and swamp the wealthiest Linux solution providers in lawsuits.

As FOSS has reached a high level of adoption in several critical markets to (among others) the Internet infrastructure, patent litigation has very concrete negative consequences on top players in these markets, as they are forced to dedicate increasing amounts of resources to litigation avoidance strategies.
3 Counter-arguments

A popular argument among software patent supporters is that without the ability to patent their inventions, software authors would have nothing to protect them from imitators building cheap knock-offs of their work. In responding to this argument, it is worth noting that patents are not the only protection mechanism available to software authors; the computer code itself is protected under copyright, the implementation is covered by trade secret law, and the brand name of the software can be trademarked. Therefore, the only aspect of a computer program potentially left unprotected by the lack of patent coverage is the functionality of the software. In other words, anybody could reverse-engineer the program and build a workalike. The pro-software patent camp argues (form the point of view of the software author) that this is bad because it discourages innovation; the anti-software camp argues (from the point of view of the consumer) that this is good because it encourages competition. There is obviously a trade-off at play here, but it is hopefully clear from the previous sections that pure software patents are not the right solution, as they tilt the balance too strongly in favor of intellectual property holders.

Software patent proponents also argue that software patents “encourage inventive efforts through the promise of economic rewards” and “provide incentives for product refinement and commercialization”. [5] The first claim is easily disproved by facts: the companies holding the most software patents (IBM, Hitachi, AT&T, DEC, Toshiba, Sharp) are widely regarded in the software industry as being “totally incapable of delivering innovative software products in the marketplace” [6]. The second claim is also debatable – considering that the software industry has had no trouble commercializing its products without patent protection for quite some time, it is unclear whether patent protection is actually needed for this purpose.

4 Conclusion

Before it was even clear that the US software industry really needed encouragement to innovate and spend money on research, legislators decided to enact public policy to this effect by way of allowing computer programs to be patented. One decade later, there is still no positive correlation between software patents and the effects they were supposed to produce. Instead, the industry has become infested with an entire zoology of patent law abusers (trivial patents, submarine patents, patent trolls), the US patent office is by their own admission stressed to the limit (the application backlog passed the 500,000 mark in 2004), and extortion runs rampant. Software patents have done little besides making patent lawyers even richer than they already were.

The American experiment with software patents has failed. Far from emulating the US patent law, the rest of the world is currently investigating alternative protection for computer programs. To get rid of abusers and restore healthy competition in the software industry, the US should join the EU in the search for the right intellectual property protection framework for computer code.
References


