Arguments against software patents in india

Centre for Internet and Society | 2010-02-22

Pranesh Prakash

CIS believes that software patents are harmful for the software industry and for consumers. In this post, Pranesh Prakash looks at the philosophical, legal and practical reasons for holding such a position in India.

This blog post is based on a presentation made at the iTechLaw conference held on February 5, 2010. The audience consisted of lawyers from various corporations and corporate law firms. As is their wont, most lawyers when dealing with software patents get straight to an analysis of law governing the patenting of computer programmes in India and elsewhere, and seeing whether any loopholes exist and can be exploited to patent software. It was refreshing to see at least some lawyers actually going into questions of the need for patents to cover computer programs. In my presentation, I made a multi-pronged case against software patents: (1) philosophical justification against software patents based on the nature of software; (2) legal case against software patents; (3) practical reasons against software patents.

Preamble

Through these arguments, it is sought to be shown that patentability of software is not some arcane, technical question of law, but is a real issue that affect the continued production of new software and the everyday life of the coder/hacker/software programmer/engineer as well as consumers of software (which is, I may remind you, everywhere from your pacemaker to your phone). A preamble to the arguments would note that the main question to ask is: **why should we allow for patenting of software**? Answering this question will lead us to ask: **who benefits from patenting of software**. The conclusion that I come to is that patenting of software helps three categories of people: (1) those large software corporations that already have a large number of software patents; (2) those corporations that do not create software, but only trade in patents / sue on the basis of patents ("patent trolls"); (3) patent lawyers. How they don't help small and medium enterprises nor society at large (since they deter, rather than further invention) will be borne out by the rest of these arguments, especially the section on practical reasons against software patents.

What are Patents?

Patents are a twenty-year monopoly granted by the State on any invention. An invention has to have at least four characteristics: (0) patentable subject matter; (1) novelty (it has to be new); (2) inventive step / non-obviousness (even if new, it should not be obvious); (3) application to industry. A monopoly over that invention, thus means that if person X has invented something, then I may not use the core parts of that invention ("the essential claims") in my own invention. This prohibition applies even if I have come upon my invention without having known about X's invention. (Thus, independent creation is not a defence to patent infringement. This distinguishes it, for instance, from copyright law in which two people who created the same work independently of each other can both assert copyright.) Patents cover non-abstract ideas/ functionality while copyright covers specific expressions of ideas. To clarify: imagine I make a drawing of a particular machine and describe the procedure of making it. Under patent law, no one else can make that particular machine, while under copyright law, no one can copy that drawing.

Philosophical Justification Against Software Patents

Even without going into the case against patents *per se* (lack of independent creation as a defence; lack of 'harm' as a criterion leading to internalization of all positive externalities; lack of effective disclosure and publication; etc.), which has been done much more ably by others like Bessen & Meurer (especially in their book Patent Failure) and Boldrin & Levine (in their book Against Intellectual Monopoly, the full text of which is available online).

But there is one essentially philosophical argument against software as subject matter of a patent. Software/computer programs ("instructions for a computer"), as any software engineer would tell you, are merely algorithms ("an effective method for solving a problem using a finite sequence of instructions") that are meant to be understood by a computer or a human who knows how to read that code.

Algorithms are not patentable subject matter, as they are mere expressions of abstract ideas, and not inventions in themselves. Computer programs, similarly, are abstract ideas. They only stop being abstract ideas when embodied in a machine or a process in which it is the machine/process that is the essential claim and not the software. That machine or process being patented would not grant protection to the software itself, but to the whole machine or process. Thus the abstract part of that machine/process (i.e., the computer program) could be used in any other machine/process, as it it is not the subject matter of the patent. Importantly, just because software is required to operate some machine would then not mean that the machine itself is not patentable, just that the software cannot be patented in guise of patenting a machine.

Legal Case Against Software Patents

In India, section 3(k) of the Patent Act reads:

> (3) The following are not inventions within the meaning of this Act: (k) a mathematical or business method or computer programme (*sic*) *per se* or algorithms.

As one can see, computer programs are place in the same category as "mathematical methods", "algorithms", and "business methods", hence giving legal validity to the idea propounded in the previous section that computer programs are a kind of algorithms (just as algorithms are a kind of mathematical method).

Be that as it may, the best legal minds in India have had to work hard at understanding what exactly "computer programme *per se*" means. They have cited U.S. case law, U.K. case law, E.U. precedents, and sought to arrive at an understanding of how *per se* should be understood. While understanding what *per se* means might be a difficult job, it is much easier to see what it does *not* mean. For that, we can look at the 2004 Patent Ordinance that Parliament rejected in 2005. In that ordinance, sections 3(k) and (ka) read as follows:

> (3) The following are not inventions within the meaning of this Act: (k) a computer programme *per se* other than its technical application to industry or a combination with hardware; (ka) a mathematical method or a business method or algorithms.

Thus, it is clear that the interpretation that "computer programme *per se*" excludes "a computer programme that has technical application to industry" and "a computer programme in combination with hardware" is wrong. By rejecting the 2004 Ordinance wording, Parliament has clearly shown that "technical application to industry" and "combination with hardware" do not make a computer programme patentable subject matter.

Indeed, what exactly is "technical application to industry"? "Technical" has various definitions, and a perusal through those definitions would show that barely any computer program can be said not to relate to a technique, not involve "specialized knowledge of applied arts and sciences" (it is code, after all; not everyone can write good algorithms), or not relate to "a practical subject that is organized according to scientific principles" or is "technological". Similarly, all software is, by definition, meant to be used in combination with hardware. Thus, it being used in combination with hardware must not, as argued above, give rise to patentability of otherwise unpatentable subject matter category.

In 2008, the Patent Office published a new 'Draft Manual Of Patent Practice And Procedure' in which it sought to allow patenting of certain method claims for software inventions (while earlier the Patent Office objected to method

claims, allowing only device claims with hardware components). This Draft Manual was withdrawn from circulation, with Shri N.N. Prasad (then Joint Secretary of DIPP, the department administering the Patent Office) noting that the parts of the Manual on sections 3(d) and 3(k) had generated a lot of controversy, and were *ultra vires* the scope of the Manual (which could not override the Patent Act). He promised that those parts would be dropped and the Manual would be re-written. A revised draft of the Manual has not yet been released. Thus the interpretation provided in the Draft Manual (which was based heavily on the interpretation of the U.K. courts) cannot not be relied upon as a basis for arguments in favour of the patentability of software in India.

In October 2008, CIS helped organize a National Public Meeting on Software Patents in which Indian academics, industry, scientists, and FOSS enthusiasts all came to the conclusion that software patents are harmful for both the industry as well as consumers.

Practical Reasons Against Software Patents

This is going to be an attempt at distilling and simplifying some of the main practical arguments against patenting of software.

There are traditionally four incentives that the patent system caters to: (1) incentive to invent; (2) incentive to disclose; (3) incentive to commercialize; and (4) incentive to invent substitutes. Apart from the last, patenting of software does not really aid any of them.

1. **Patent Landmines / Submarine Patents / Patent Gridlocks / No Exception for Independent Creation**

   Given that computer programs are algorithms, having monopolies over such abstract ideas is detrimental to innovation. Just the metaphors say a lot about software patents: landmines (they cannot be seen/predicted); submarines (they surface out of the blue); gridlocks (because there are so many software patents around the same area of computing, they prevent further innovation in that area, since no program can be written without violating one patent or the other).

   Imagine the madness that would have ensued had patents been granted when computer programming was in its infancy. Imagine different methods of sorting (quick sort, bubble sort) that are part of Computer Science 101 had been patented. While those particular instances aren't, similar algorithms, such as data compression algorithms (including the infamous LZW compression method), have been granted patents. Most importantly, even if one codes certain functionality into software independently of the patent holder, that is still violative of the patent. Computer programs being granted patents makes it extremely difficult to create other computer programs that are based on the same abstract ideas. Thus incentives # (1) and (3) are not fulfilled, and indeed, they are harmed. There is no incentive to invent, as one

would always be violating one patent or the other. Given that, there is no incentive to commercialize what one has invented, because of fear of patent infringement suits.

An apt illustration of this is the current difficulty of choosing a royalty-free video format for HTML 5, as it shows, in practical terms, how difficult it is to create a video format without violating one patent or the other. While the PNG image format was created to side-step the patent over the LZW compression method used in the GIF image format, bringing Ogg Theora or Dirac (both patent-free video format) to surpass the levels of H.264/MPEG-4 AVC or VC-1 will be very difficult without infringing dozens if not hundreds of software patents. Chris DiBona of Google, while talking about improving Ogg Theora as part of its inclusion in HTML 5 specifications said, "Here's the challenge: Can Theora move forward without infringing on the other video compression patents?" Just the number of companies and organization that hold patents over H.264 is astounding, and includes: Columbia University, Electronics and Telecommunications Research Institute of Korea (ETRI), France Télécom, Fujitsu, LG Electronics, Matsushita, Mitsubishi, Microsoft, Motorola, Nokia, Philips, Robert Bosch GmbH, Samsung, Sharp, Sony, Toshiba, and Victor Company of Japan (JVC). As is the amount of royalties to be paid ("[t]he maximum royalty for these rights payable by an Enterprise (company and greater than 50% owned subsidiaries) is $3.5 million per year in 2005-2006, $4.25 million per year in 2007-08 and $5 million per year in 2009-10"; with royalty per unit of a decoder-encoder costing upto USD 0.20.)

Indeed, even the most diligent companies cannot guard themselves against software patents. FFII estimates that a very simple online shopping website would violate twenty different patents at the very least. Microsoft recently lost a case against i4i when i4i surfaced with a patent covering custom XML as implemented in MS Office 2003 and MS Office 2007. As a result Microsoft had to ship patches to its millions of customers, to disable the functionality and bypass that patent. The manufacturers of BlackBerry, the Canadian company Research in Motion, had to shell out USD 617 million as settlement to NTP over wireless push e-mail, as it was otherwise faced with the possibility of the court shutting down the BlackBerry service in the U.S. This happened despite there being a well-known method of doing so pre-dating the NTP patents. NTP has also filed cases against AT&T, Sprint Nextel, T-Mobile, Verizon Wireless, and Palm Inc. Microsoft was also hit by Visto Corporation over those same NTP patents, which had been licensed to Visto (a startup).

- **Don't These Cases Show How Software Patents Help Small Companies?**

  The astute reader might be tempted to ask: are not all of these examples of small companies getting their dues from larger companies? Doesn't all of this show that software patents actually help small and medium

enterprises (SMEs)? The answer to that is: no. To see why, we need to note the common thread binding i4i, NTP, and Visto. None of them were, at the time of their lawsuits, actually creating new software, and NTP was an out-and-out "non-practising entity"/"patent holding company" AKA, patent troll. i4i was in the process of closing shop, and Visto had just started up. None of these were actually practising the patent. None of these were producing any other software. Thus, none of these companies had anything to lose by going after big companies. In other words, the likes of Microsoft, RIM, Verizon, AT&T, etc., could not file counter-suits of patent infringement, which is normally what happens when SMEs try to assert patent rights against larger corporations. For every patent that the large corporation violates of the smaller corporation, the smaler corporation would be violating at least ten of the larger corporation's. Software patents are more helpful for software companies as a tool for cross-licensing rather than as a way of earning royalties. Even this does not work as a strategy against patent trolls.

Thus, the assertion that was made at the beginning is borne out: software patents help only patent trolls, large corporations that already have large software patent portfolios, and the lawyers who draft these patents and later argue them out in court.

2. **Term of Patents**

   Twenty years of monopoly rights is outright ludicrous in an industry where the rate of turnover of technology is much faster – anywhere between two years and five months.

3. **Software Industry Progressed Greatly Without Patents**

   In India, software patents have never been asserted in courts (even though many have been illegally granted), yet the software industry in India is growing in leaps and bounds. Similarly, most of the big (American) giants of the software industry today grew to their stature by using copyright to "protect" their software, and not patents.

4. **Copyright Exists for Software**

   As noted above, the code/expression of any software is internationally protected by copyright law. There is no reason to protect the ideas/functionality of that software as well.

5. **Insufficient Disclosure**

   When ordinary computer programmers cannot understand what a particular software patent covers (which is the overwhelming case), then the patent is of no use. One of the main incentives of the patent system is to encourage gifted inventors to share their genius with the world. It is not about gifted inventors paying equally gifted lawyers to obfuscate their inventions into gobbledygook so that other gifted inventors can at best hazard a guess as

to precisely what is and is not covered by that patent. Thus, this incentive (#2) is not fulfilled by the current system of patents either – not unless there is a major overhaul of the system. This ties in with the impossibility of ensuring that one is not violating a software patent. If a reasonably smart software developer (who are often working as individuals, and as part of SMEs) cannot quickly ascertain whether one is violating patents, then there is a huge disincentive against developing software in that area at all.

6. **Software Patents Work Against Free/Libre/Open Source Software**

   Software patents hinder the development of software and FOSS licences, as the licensee is not allowed to restrict the rights of the sub-licensees over and above the restrictions that the licensee has to observe. Thus, all patent clearances obtained by the licensee must be passed on to the sub-licensees. Thus, patented software, though most countries around the world do not recognize them, are generally not included in the default builds of many FOSS operating systems. This inhabits the general adoption of FOSS, since many of the software patents, even though not enforceable in India, are paid heed to by the software that Indians download, and the MP3 and DivX formats are not enabled by default in standard installations of a Linux OS such as Ubuntu.

Conclusion

Currently, the U.S. patent system is being reviewed at the administrative level, the legislative level, as well as the judicial level. At the judicial level, the question of business method patents (and, by extension, software patents) is before the Supreme Court of the United States of America in the form of *Bilski v. Kappos*. Judge Mayer of the Court of Appeals for the Federal Circuit (CAFC, which heard *In re Bilksi*) noted that "the patent system has run amok". The Free Software Foundation submitted a most extensive *amicus curiae* brief to the U.S. Supreme Court, filled with brilliant analysis of software patents and arguments against the patentability of software that is well worth a read.