

**BRIEFING NOTE ON THE IMPACT OF SOFTWARE PATENTS ON
THE SOFTWARE INDUSTRY IN INDIA**

PREPARED BY

**LAWRENCE LIANG
ANURANJAN SETHI
PRASHANTH IYENGAR**

Background

While there has been a lot of discussion on the impact that the latest amendment to the Indian Patent Act will have on public health and the pharmaceutical sector in India, there has been a disturbing silence about the impact that the amendment has on the software industry.

After the patents (second amendment) in 2002, the scope of non patentable subject matter in the Act was amended to include the following: “a mathematical method or a business method or a computer programme per se or algorithms”.

The important phrase that was added was ‘per se’, and with the amendment we effectively included Software patents into Indian Law. The latest amendment seeks to expand the scope of software patents, and states “a computer programme per se other than its technical application to industry or a combination with hardware; a mathematical method or a business method or algorithms”.

This briefing note will address the larger question of why we should be concerned about software patents, and the impact that it will have on the software industry in India.

I. Conceptual difference between Copyright and Patent

The first thing to note is that software is already protected under Copyright law, so what then is the motivation and the implication of a move from copyright protection to patent protection?

Software has traditionally been protected under copyright law since code fits quite easily into the description of a literary work. Software Patenting has recently emerged (if only in the US, Japan and Europe) as an alternative that software companies are increasingly employing to, in order to protect their products.

The issues involved in conferring patent rights to software are, however, a lot more complex than taking out copyrights on them. Specifically, there are two challenges that one encounters when dealing with software patents. The first is about the **instrument of patent** itself and whether the manner of protection it confers is suited to the software industry. The second is the **nature of software**, and whether it should be subject to patenting.

a) Different Subject Matters

Copyright protection extends to all original literary works (among them, computer programs), dramatic, musical and artistic works, including films. Under copyright, protection is given only to the *particular expression* of an idea that was adopted and not the idea itself. (For instance, a program to add numbers written in two different computer languages would count as two different expressions of one idea) Effectively, *independent rendering* of a copyrighted work by a third party would not infringe the copyright.

Generally patents are conferred on any **'new' and 'useful'** art, process, method or manner of manufacture, machines, appliances or other articles or substances produced by manufacture. Worldwide, the attitude towards patentability of software has been skeptical. The Indian Patent Act, as modified in 2002 had made non-patentable the following:

"...a mathematical method or a business method or a computer programme per se or algorithms".

However, the recent amendment ordinance states instead:

*"...a computer programme per se other than its technical application to industry or a combination with hardware;
a mathematical method or a business method or algorithms."*

b) Who may claim the right to a patent/copyright?

Generally, the author of a literary, artistic, musical or dramatic work automatically becomes the owner of its copyright.

Software developers are perfectly protected without patents. Everyone who writes a computer program automatically owns the copyright in it. It's copyright law that made

Microsoft, Oracle, SAP and the entire software industry so very big. It's the same legal concept that also protects books, music, movies, paintings, even architecture.

Many of the world's richest people owe their wealth to copyright law. Some examples are: Bill Gates, Paul Allen and Steve Ballmer (Microsoft); Larry Ellison (Oracle); Hasso Plattner and the other founders of SAP; Paul McCartney (Beatles); JK Rowling (Harry Potter).

The patent, on the other hand is granted to the *first to apply for it*, regardless of who the *first to invent it* was. Patents cost a lot of money. They cost even more paying the lawyers to write the application than they cost to actually apply. It takes typically some years for the application to get considered, even though patent offices do an extremely sloppy job of considering.

c) Rights conferred

Copyright law gives the owner the **exclusive right** to reproduce the material, issue copies, perform, adapt and translate the work. However, these rights are tempered by the rights of *fair use* which are available to the public. Under “fair use”, certain uses of copyright material would not be infringing, such as use for academic purposes, news reporting etc. Further, independent recreation of a copyrighted work would not constitute infringement. Thus if the same piece of code were independently developed by two different companies, neither would have a claim against the other.

A patent confers on the owner an **absolute monopoly** which is the the right to prevent others from marking, using, offering for sale without his/her consent. In general, patent protection is a far stronger method of protection than copyright because the protection extends to the level of the idea embodied by a software and injuncts ancillary uses of an invention as well. It would weaken copyright in software that is the base of all European software development, because independent creations protected by copyright would be attackable by patents

Many patent applications cover very small and specific algorithms or techniques that are used in a wide variety of programs. Frequently the "inventions" mentioned in a patent application have been independently formulated and are already in use by other programmers when the application is filed.

d) Duration of protection

The TRIPS agreement mandates a period of at least 20 years for a product patent and 15 years in the case of a process patent.

For Copyright, the agreement prescribes a minimum period of the *lifetime of the author plus seventy years*.

II. Nature of Software and Indian Software Industry

Software is complex: The complexity of computer programs makes it difficult to be understood by any one person. This capacity for complexity allows for the creation of highly sophisticated products but also means that they are **dependent on a vast range of technologies**.

Software is free from the constraints of the real world that ensure a product does not become too complex. Major software may comprise up to 10 million lines of code - potentially *thousands of inventions*, any of which might be patented

For example, Apple was sued because its HyperCard program allegedly violates patent number 4,736,308, which covers a specific technique that, in simplified terms, entails scrolling through a database displaying selected parts of each line of text. Separately, the scrolling and display functions are ubiquitous fixtures of computer programming, but combining them without a license from the holder of patent 4,736,308 is now apparently illegal.

In its complexity, software is different from other engineering and mechanical inventions for which patent protection was devised. The latter are often characterized by large "building block" inventions that can revolutionize a given mechanical process. Software, especially a complex program, seldom includes substantial leaps in technology, but rather consists of adept combinations of many ideas. Whether a software program is a good one does not generally depend as much on the newness of a specific technique, but instead depends on the unique combination of known algorithms and methods. Patents should not protect such methods of innovation.

Software Technology evolves rapidly: Software technology is evolving much faster than other industries, even with its own hardware industry. Against this light, a patent that lasts upto 17 years is extremely alarming. Microprocessors double in speed every 2 years.

Research in software is galloping ahead of developments. In most industries, researching new ideas often costs more money than bringing them to the market. The software industry is, on the other hand, loaded with ideas.

The idea behind most software patents can be coded in just 20 lines of code, but any program incorporating that idea - along with many others - will be a thousand times larger. It is the *writing of a program* that takes all the time, not *coming up with ideas*.

What this means is that on an average of every two years, a *product* will have to be replaced in the market. The *idea* underlying it will remain the same although the particular means and variants of its applications may have changed radically.

Coming out with a full-featured product, every two years is costly especially in relation to the inexpensive idea that backs it. There's more novelty in the development and application of the same idea to new technology than with coming up with the original raw idea.

The objective of granting patent rights should be to foster the growth and evolution of the industry. Granting a patent at this stage would be akin to unreasonably prolonging the life of a product.

It is generally found that those who are investing time creating and lodging patents are vastly outpacing those who are investing effort bringing such ideas to market. By the time an immature technology develops to the point where it can be incorporated into products, it has a dozen or more patents on it that render it commercially intractable.

Software doesn't wear out: In other industries, research continues up to a point where further research costs too much to be feasible. At this stage, the industry's output merely consists of replacing parts that have worn out.

However, in the software sector, a computer program that is fully debugged will perform its function forever without requiring maintenance or modification. “What this means is that unlike socks that wear out, and breakfast cereal that is eaten, a particular software product can be sold to a particular customer at most once. If it is to be sold to that customer again, it must be enhanced with new features and functionality.” This inevitably means that *even if the industry were to approach maturity, any software company that does not produce new and innovative products will simply run out of customers!* Thus, the industry will remain innovative whether or not software patents exist.

Software has different economics: Most other major industries have medium to high research and development costs and very high production costs. Most often, the production costs dwarf the other two areas (because of the physicality that they involve) so that these costs can be added on to the cost of the final product without any relatively major difference in the price.

Software is unique in this aspect because

-The research costs very little because “ideas are as abundant as air”

-The development of an idea into a marketable product costs far more than the research.

-The production costs are minimal, often just a little more than the price of the medium, which is typically a floppy or a CDROM.

Patents affect the ‘development’ stage of the process of ‘manufacture’ of software. Thus the threat exists that the price of software could be singularly determined by the number of patented innovations that it incorporates.

III. Patent and Innovation in Software Industry

As argued before the process of software development by its very nature is 'incremental' i.e. developing of new software majorly consists of building upon existing ideas and rearranging the processes devised by others, and hence has an inbuilt need for using existing algorithms and mathematical formulae. Patent protection over software or over a set of algorithms within patented software would inevitably create a thicket of patents which the subsequent software developer might need to obtain clearance from before he can begin to work on it. The costs involved in obtaining these clearances and those involved in case one finds oneself having infringed a patent are usually very high, as in the case of biomedical patents. This would act as a disincentive for an aspiring software developer and would adversely affect the growth of the Indian software industry. Introduction of two bills- '*Genomic Research and Diagnostic Accessibility Bill, 2002*' and '*Genomic Science and Technology Innovation Act of 2002*' though still pending before the US Congress show the real concerns involved for a 'patent and innovation policy' within genomics. Similar concerns are exist in the software and innovation policy and need to be addressed adequately by the each national legislature.

Further there are substantial costs involved in verifying which patents one must obtain clearance for as skimming through the huge patent databases has become a very costly exercise. Unfortunately, conducting a patent search is a slow, deliberative process that, when harnessed to software development, could stop innovation in its tracks. And because patent applications are confidential, there is simply no way for computer programmers to ensure that what they write will not violate some patent that is yet to be issued making survival a very important issue for smaller player in the market.

Various large companies in US have obtained exemptions from going through patent searches for standard work due to huge costs. In such a scenario in a small player software industry like India, it would be unwise to allow 'software patents' as they may have negative impact upon the innovation within the industry.

By its nature software industry is 'innovation driven' i.e. the only way a software company can compete and improve its sales or grip over market is by making better and more useful features available. This innovation which is the driving force behind the Indian software industry is bound to get affected if a patent protection is provided to software patents. If a company can easily sustain itself on its 'invention' (by obtaining patents upon its software) and need not remain innovation driven, which would mean that a patent monopoly would inversely impact innovation and competition in software industry. It would further give rise to monopolistic tendencies and a practice of quoting arbitrary price for the grant of 'voluntary license'. This lesson can be learnt by looking west where the idea of Public Key Encryption was patented in the US. The patent expired in 1997 and until then, it largely blocked the use of Public Key Encryption in the US. Similar instances can be found w.r.t. 'data compression software' and 'single click software' patented by Amazon.com. A number of programs that people started to develop got crushed. They were never really available because the patent holders threatened them. This led to a lot of unrest in the software community which culminated into the public outrage against software patents. Similar pressures have prevailed in European community where software patents found public opposition too immense to mount for a long time.

A look at India's own development of its software industry would be of immense help as India started its software industry only after IBM was driven out of country. Before that, there was no software industry worth the name, with software and hardware being imported from IBM. Once IBM left, Indian computer companies developed computers using the UNIX operating system, which was in the public domain. This led to the presence of a large number of skilled software professionals with experience of UNIX were also writing high-level applications for making the entire computer system work.

IV. Political economy of Software Patents

While understanding the issue of software patents, it's important to look at its political economy and the implications involved for India. If one were to study the trends of software patenting in US and Europe one would witness that the IBM owns a majority of patents along with other giant software companies and has been topping the list of maximum patents granted in US in the private sector. This fact must be seen in the light of the opposition faced from small business organisations, leading scientists and economists in Europe and the unprecedented delay in passing the Software Patent Directive of 2002 by the European parliament. It should be noted that the directive does not aim to make it possible to patent pure computer programs: it would only apply to computer software integrated into an appliance. This makes it much more restrictive than the amended Indian Patent Act, which opens out any technical application of a programme to industry or its realisation in hardware for patenting. Even with this restriction, the critics of the EU directive have pointed out that a patent on software is in effect a patent on an idea, while traditionally patents have been restricted to concrete physical inventions only. By making this amendment, it is possible to implement algorithms in hardware and then claim patent protection for this. Once an idea can be patented if it is burnt in to hardware, the argument for extending it to a software implementation gains ground. In fact, the first breach in the US for making software patentable came through this route. If one were to study the trends in the scope of patentable subject matter granted in software patents by US courts, one would observe that from *Diamond v. Diehr* onwards court has been granting patents on much more abstract components, which has slowly transformed into patenting the central idea underlying the software. This trend indicates the easy malleability of legal terminology which has brought US courts' stand on software patents to a full circle from *Gottschalk v. Benson* where the court found a patent upon software as a patent upon the underlying algorithms which is nothing more than a mathematical formula, unpatentable by its very definition.

The concerns regarding the weaker relative position of these small players is much more relevant in India. Among the primary reasons for large corporations like IBM lobbying for software patents is due to their stronger hold over the software market and ownership of the largest number of patents in this market. Large corporations use their patents, apart from making royalty upon them, to getting access benefit to the patents of other companies. This would close the option of cross-licensing for a majority of Indian companies which have no patents upon software. License though may be obtained are usually available at exorbitantly high prices which would most likely be unaffordable for Indian companies which operate on a small scale and have restricted budget options. The multinational corporations would use software patents as a defensive strategy for preventing smaller Indian companies from gaining any grounds in the market, which would eventually drive them out of business hence destroying the existing Indian software industry.

Software industry has a very characteristic nature which makes it extremely vulnerable to being easily monopolized. Among these characteristics are *Network effects* (the fact that a program becomes more useful if more people use it), *interoperability and compatibility problems*, the *low cost of massive reproduction of software*, the *difficulty of*

inspecting software distributed without the source code, the learning curve and the rapid evolution of the market. Taking the instance of Microsoft Windows (the most popular operating system in use in India today) which enjoys a perpetual monopoly over the operating system market in India, many a larger institutions find Windows extremely costly and desperately needed an alternative to it in order to do business profitably. The recent success of Linux operating systems is demonstrative of this, but this must be understood in the light that India follows a copyright regime for software which allows many of the above mentioned characteristics of compatibility and interoperability to be resolved which would be totally impossible in a software patent regime. This then means that software patents have a potential to hamper the growth of open software movement in India which has begun to play central role in Indian Government's 'e-governance' initiative. Hence it's extremely urgent to ensure that patents in software do not cause any harm to the fine balance that copyright has achieved.

While understanding a political economy argument of software patents the adverse impact of monopolization upon public interest which has been held to be of utmost importance by the apex court in India, even above one's legitimate commercial interests.

V. Procedural Issues

There are a certain procedural issues involved which are of determinative nature as to the allowance of a software patent regime in India. India doesn't have a well laid out or even a well practiced software patent practice to guide Indian patent office. In the absence of any such policy, examining software patent application becomes a very daunting task, coupled with which the complicated and highly technical nature of software, Indian patent office is quite incapable to evaluate complicated and technically trivial claims which software patent often present. Imposing a software patent regime in such a scenario would impact the quality of such patents which might then prove counter-productive in the development of Indian software industry.

To be able to tackle this situation more personnel and experts would have to be employed in the patent office that can then ensure maintenance of a certain quality standards while granting software patents. But this in turn may not produce increased innovation in the software industry for the human capital which would be invested into processing the claims and preventing and tackling with the patent infringements rather than being invested in developing new software and hence benefit the software industry and economy of the country in general.

The difficulty in reaching a policy to grant software patents and the impacts of granting these patents in the absence of policy are indeed far reaching. In the absence of a policy which classifies patents on algorithms, techniques etc. it would take an awfully long time for the patent office to process a claim, searching the 'prior art' which makes the system inefficient and unworkable. Long delays in processing patent applications and subsequent challenge procedure often makes filing for a patent an unwise option for small companies and individual software developer, which form the backbone of Indian software industry. For instance, IBM was granted a patent on the same data-compression algorithm that Unisys supposedly owned. Such an error which could prove lethal for a developing company which has planned its budget meticulously and in consequence of this error would be greatly disincentivized to develop new software. The Patent Office was probably not aware of granting two patents for the same algorithm because the descriptions in the patents themselves are quite different even though the formulas are mathematically equivalent. Even when patents are known in advance, software publishers have generally not licensed the algorithms or techniques; instead, they try to rewrite their programs to avoid using the particular procedure that the patent describes. Sometimes this isn't possible, in which case companies have often chosen to avoid implementing new features altogether. It seems clear from the evidence of the last few years that software patents are actually preventing the adoption of new technology, rather than encouraging it.