

Software Patents: Background, Discussion and Illustration of Harm
(c) CC-BY, 23rd April 2006, Ditesh Kumar

1. Introduction

A patent is the legal recognition of an invention that satisfies certain criteria. A patent grant would award the patentee a monopoly right for a limited number of years. The purpose of a patent is twofold:

- to encourage new inventions by recognizing the inventor
- to allow the patentee to benefit from commercial exploitation of the invention (by making, selling, using or importing).

Patents play a very important role in increasing the intellectual value of the knowledge in society. To allow a patent to be awarded, the patentee has to fulfill his or her obligation by specifying in detail the full specifications of the invention. This information is openly published (by the patents office) so that the public may view the disclosure and benefit from it (for example, by using it in further research and development or licensing the patent for use).

1.1 Qualification

A patent has to satisfy three criteria:

1. **Novelty:** the invention must be new and not be already published, used or be part of the state of art. For the patent request to be valid, the invention must not have been used by the public anywhere in the world prior to the filing of the application.
2. **Non-obvious:** the invention must represent an important advancement in prior art.
3. **Capable of industrial application**

2. Software Patents

There has been a vigorous discussion on patents for software methods and algorithms. Software patents are relatively new and its introduction has been fiercely opposed by various parties, including academicians, companies and professional bodies. While the United States Patent and Trademark Office (USPTO) and the Supreme Court of the United States have explicitly allowed for software patents, many other countries do not have specific legal rules with regards to patenting software methods and algorithms although there is currently efforts to clarify the issue.

2.1 Software Patents in the United States

Patents were traditionally not approved for mathematical techniques anywhere in the world as only physical processes were given consideration. The USPTO clearly stated that the

following materials could not be patented as they were indistinguishable from abstract ideas and laws of nature:

- data structures or programs per se (which are mere information rather than a computer implemented process or specific machine or computer readable memory as an article of manufacture);
- compilations or arrangements of non-functional information or a known machine-readable storage medium encoded with such information; or

Computer software was not considered patentable before the U.S. landmark case, *Diamond v. Diehr*, in which the U.S. Supreme Court ordered the patent office to grant a patent on a computer software invention. The U.S. Supreme Court's original stance was to disallow software patents. In *Gottschalk v. Benson*, the Supreme Court had to decide whether a patent could be granted for a mathematical formula whereby the use of the mathematical formula was not limited to an art or technology, a particular apparatus or a particular end use. The use of the formula was in a general-purpose computer. The Supreme Court denied the request and held that the formula did not have a practical application except when used with a computer, and by allowing a patent the implication would be that the patent is on the algorithm; the decision prevented the patenting of software methods.

However, 10 years later, the Supreme Court changed its opinion and granted a software patent in the *Diamond v. Diehr*. In this case, the invention for which the patent was awarded described a method to heat rubber such that the rubber is best "cured." The novel part of the invention was that a computer was used to calculate and control the timing for the heating of the rubber and as the patent not only included the use of the computer but also the steps of heating the rubber, the U.S. Supreme Court allowed for the patent stating that the invention was a process for molding rubber and not just a mathematical algorithm.

Given this decided case, the lower courts attempted to set guidelines as to which inventions were mere mathematical algorithms and which inventions are worthy of a patent. Given that it is very difficult to separate an abstract mathematical idea from a mathematical idea that can be applied in practice, the Federal Circuit attempted to clarify when a software related invention can be considered patentable:

1. The entire invention (program) should be examined, and if the invention or program is only a mathematical algorithm, then the invention cannot be patented.
2. However, if the invention makes use of numbers in such a manner that concrete, real world values are computed, then the invention can be considered patentable. Real world values is referred to values that is determined from outside the computer, for example by receiving input from an outside machinery.

Another decided case was *Alappat vs U.S.P.T.O.* where Kurippan Alappat, Edward E. Averill and James G. Larsen (collectively to be referred to as Alappat) applied for a patent for their invention of creating a waveform display on a digital oscilloscope.

The patent claim was summarily rejected because the claim was on a nonstatutory subject matter (a mathematical operation, in this case) and the Patent Board ruled that '*when the claim is viewed without the steps of this mathematical algorithm, no other elements or steps are found*'. In Malaysia as well as the U.S., non statutory subject matter include laws of nature (physics, chemistry and biology), natural phenomena (such as Maxwell's theory of

electromagnetics) and abstract ideas (such as mathematical concepts and theorems including geometry, calculus etc). Specific to the U.S., section 101 of their Patent Act states that:

“Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.”

The U.S. Supreme Court made it clear that the non statutory subject matter could be patented, and the provisions given Section 101 excludes the subject matter. In *Gottschalk v. Benson*, the Supreme Court defined an algorithm as:

“a procedure for solving a given mathematical problem, in a generalized or abstract form such that [f]rom the generic formulation [computer] programs may be developed as specific applications.”

This definition was used by the U.S. Court of Appeals for the Federal Circuit to note that subject matter that includes mathematics and/or algorithms can be patented only if it can be used in a practical application, and that the subject matter itself is not entitled to patent protection. Thus, in the appeal procedure for *Alappat vs U.S.P.T.O.*, the Federal Circuit ruled that the while the subject matter could not be patented, the patent involved a practical application of the subject matter, and thus the overall invention could be patented.

This was a dangerous precedent as Alappat's invention was, in essence, a conversion of the data by means of a mathematical operation. This conversion is generally carried out in a large body of computer programs and is obvious to practitioners of the art. With the grant of the patent, the Federal Circuit effectively rejected the notion a general purpose computer could never be patented. The ruling stated that if the general purpose computer is programmed to run a specific function, then the computer becomes a special purpose computer.

In the dissent opinion, the Chief Judge stated the Supreme Court judgment was flawed because:

“Mathematical operations, like ideas and laws of nature, are not useful applications and therefore not statutory subject matter. The hypertechnical distinction between calling something a mathematical ‘algorithm’ versus another mathematical noun is without legal distinction.”

The Chief Judge stated that the rasterizer (the tool used to convert the data) is a mathematical idea that *“as a whole thus claims old circuitry elements in an arrangement defined by a mathematical operation which only performs the very mathematical operation that defines it”* . Thus, the Chief Judge found the claim not patentable as the invention was of non statutory matter. With these two cases, the USPTO drew up a set of guidelines for patent examiners to follow.

In 1998, the Federal Circuit of the United States allowed for a patent for a financial system that automatically calculated and allocated profits from a joint stock account (*State Street Bank & Trust v. Signature Financial Group*). After this point, software patents were explicitly allowed and the number of software patents granted in the U.S. rose to more than 30,000.

2.2 Software Patents in Europe

The European Patent Office (EPO) originally stated that programs for computers and presentations of information could not be patented. Article 52 of the European Patent Convention of 1973 laid down rules excluding mathematical methods, intellectual methods, business methods, computer programs, presentation of information etc as inventions. The EPO stated that:

“If the contribution to the known art resides solely in a computer program then the subject matter is not patentable in whatever manner it may be presented in the claims. For example, a claim to a computer characterized by having the particular program stored in its memory or to a process for operating a computer under control of the program would be as objectionable as a claim to the program per se or the program when recorded on magnetic tape.”

In 1985, the guidelines were reviewed because increasing number of software companies petitioned for software patents. The subsequently reviewed patent policy stated that while computer programs per se were not patentable, if the subject matter increased the technical knowledge in the relevant field(s), then patentability should not be denied merely on the ground that a computer program was involved in its implementation. Technically, these were process claims as the claim was on the process and not on the software itself. In 1998, the EPO introduced program claims which effectively allowed patents on the software.

The revised guidelines were used in the patent application for Vicom Systems' Application where the appellants applied for a patent which described a method of digitally processing images using a computer. The patent board decided that even though the idea was abstract and was mathematical in nature, the claim was not on the mathematical method but on the application of this method in a computer based system. The patent board drew a distinction where a method of digitally filtering data was considered a mathematical method and producing a rearranged data array was considered sufficiently non mathematical. From this decision, further applications for patentability of software related inventions have been dealt by considering whether there still a “technical problem” to be solved once the program element has been removed from the claim.

There needs to be clarification as to the nature of a “technical problem”. From the various decisions made, a technical problem is informally defined as a problem that cannot be solved mentally. For example, the process of translating printer control features in a word processing program to another program has been considered as a solution to a technical problem. On the other hand, cases whereby the claim was rejected due to the non-technical nature of the patent was a claim by IBM for their text processing system. The program could detect and correct contextual homophone errors but the patent board ruled that this act was purely mental. In this vein, another problem that was considered to be solved by a mental process was determining the form of a letter depending on the position of the letter in the sentence.

It is difficult to derive from the cases any general test that would determine the exact nature of a technical question. It can be inferred from the various cases that the manipulation of raw digital information is fine but if the manipulation requires semantic analysis of the text, then the patent is not to be allowed. For example, Siemens/Character and Vicom both claimed for processes that allowed the alteration of digital files. The Siemens case was

rejected because the claim was based upon the rules of writing and required the text to have meaning, whereas the Vicom claim was approved because the method for which the patent was granted could be applied to any image.

Further cases in United Kingdom make the patent process for software clearer. In *Merrill Lynch's Application* [1989] RPC 561, the patent claim was for a computer system where a trading market in securities could be set up via a computerized method which would then be programmed using known techniques. The Court of Appeal held that the invention could not be excluded simply because it used a computer but it rejected the patent claim nonetheless because the claim was a method of conducting business. In *Gale's Application* [1991] RPC 305, Aldous J held that the computer program which was held on a ROM chip could be patented. The computer program in this case only calculated square roots of numbers. The Court of Appeal reversed this decision and stated that differences in the physical structure of a program is immaterial.

In 2002, the European Commission's Directorate for the Internal Market submitted a proposal for a directive to harmonize state laws in Europe and had the alleged aim of preventing EPO from overly broad patent grants. However, reading of the actual text reading affirmed that the Commission had proposed to codify the current patent practices of EPO with the exception of program claims. In 2003, the European Parliament voted to incorporate amendments to the directive that would clearly and specifically outline the un-patentability of software and business methods. As per procedures, this amended directive was examined by the Council of Ministers' Working Party. The Working Party, as the result of secret negotiations, produced a "compromise document" that reinstated the original Commission proposal with the additional inclusion of program claims. In 2004, the Council of Ministers approved the Working Party's text by a slim majority.

The Council's decision needed to be ratified at the second stage. However, after the session in the Council, the Polish government stated that their government did not support the Working Party's position. The Dutch Parliament followed with vote of distrust against their government, which had allegedly misinformed its Parliament and asked its government to withdraw its original support. In Germany, its Parliament voted for a motion criticizing the council text and asked for changes "in the spirit of the Parliament's first reading". In the following months, there were various attempts to have the directive implemented without due process being followed, although all these attempts failed.

In April 2005, the European Parliament conducted a second reading on the Council's "Uncommon Position", as it was colloquially referred to. The text was rejected by the European Parliament on July 6, 2005.

2.2.1 Comparison of the European Approach with the United States Approach

The European approach to patentability of software is essentially determined by the contribution of the claim to the technical knowledge. According various text, patent claim drafters have been adept at framing their claims in such a manner that claims are not refused because they are computer programs. However, many such claims are refused when prior art is considered.

Patent law for software in the United States is different as the application of a formula is explicitly allowed. The new USPTO guidelines state that if computer-readable memory

influence the way a computer process is carried out, then the patent claim can be awarded. The European approach is much cautious and requires more detail on the nature of the claim.

2.3 Software Patents in Japan

Besides Europe and the U.S., Japan is the only other country who is a net exporter of intellectual rights. The patent laws in Japan are much lenient when considering software patents. This is due to Article 2(1) which defines the nature of an invention as "*the highly advanced creation of technical ideas utilizing natural laws*". This definition is limited by Article 29 which states that an invention must be useful in the industry and Article 32 states that certain types of inventions cannot be granted patent protection.

In 1971, the Japanese Patent Office issued examination standards for program related inventions. These standards were revised in 1993. Generally, the guidelines are similar to those issued by the European Patent Office in the sense that the guidelines try to determine whether a technical problem has been solved by the patent claim. The guidelines are less restrictive compared to the European guidelines because the Japanese guidelines attempt to follow in the spirit of Article 2(1) which is quite broad. The Japanese guidelines state that the following types of claims can be patented:

1. The utilization of a law of nature in information processing performed by the software (for example in computer control of apparatus used for other purposes, operations controlling the computer itself, video image processing, transmission error detecting and method of generating and displaying certain symbols).
2. Inventions using hardware resources (examples include a command input method by higher hierarchical menu selection and methods of converting Japanese phonetic letters into Chinese characters).

As these guidelines are more lenient than that of the European Patent Office, the first and second guidelines allow for practically a wide range of software patents. Many patents that would not have been allowed in Europe would be allowed in Japan. Hence, it is not surprising that Japan has the second highest number of registered software patents, second only to the United States.

2.4 Software Patents in India

The Indian Parliament rejected a clause to allow software patents in April 2005.

2.5 Patent Law in Malaysia and Its Application to Software Patents

The current Malaysia patent law does not have any specific provision for software related inventions. Section 13(1)(a) of the Malaysian Patent Act 1983 specifically excludes from patentable subject matter "*discoveries, scientific theories and mathematical methods*" and paragraph (c) excludes "*schemes, rules or methods for doing business, performing purely mental acts or playing games*". This is fairly similar to the patent law in UK as describe above.

Malaysia has adopted a modified system in its examination of patent applications that depend on whether the claims have been registered in other countries. In this manner, software based patents have been granted in Malaysia. For example, patent grant number MY-100183-A is a patent for a traffic data system where traffic data is collection from various exchanges for storage and updates. This patent was granted to NEC Corporation in Malaysia and its grant was based on the fact that the same patent had already been issued in the United States.

There is little to comment on the Malaysian Patent Act 1983 for patent applied in Malaysia for software inventions because few patents have been granted that way. It would seem that Malaysia is applying the patent guidelines as it occurs in Europe. Patent legislation in Malaysia states that the registration of a patent is effective for the whole of Malaysia. The patent act also states that the patent length is 15 years after the date of its grant. The patent act also states that the owner of a patent has the right to exploit the patented invention, to assign or transmit the patent, and to conclude license contracts.

2.6 TRIPS Provisions

TRIPS ensures minimal rules for national intellectual rights so that nations cannot use intellectual rights as a hidden trade barrier against other nations. Article 27 of TRIPS has been used to imply that patent claims must be allowed for computer programs. That article states that:

"Subject to the provisions of paragraphs 2 and 3, patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application."

Subject to paragraph 4 of Article 65, paragraph 8 of Article 70 and paragraph 3 of this Article, patents shall be available and patent rights enjoyable without discrimination as to the place of invention, the field of technology and whether products are imported or locally produced."

This has been interpreted by certain parties to indicate that TRIPS compliance requires software patents to be granted. However, the German court and the European parliament does not see TRIPS compliance as requiring an expansion of patentable subject matter. The German Federal Court stated that:

"The Treaty on Trade Related Aspects of Intellectual Property Rights (TRIPs) does not entail any different judgment of patentability. Independently of the question as to in what form - directly or indirectly - the TRIPs treaty is applicable here, the application of Art 27 TRIPs would not lead to any extension of patentability here. The wording, according to which patents shall be available for inventions in all fields of technology, merely confirms the dominating view of German patent jurisprudence, according to which the concept of technology (Technik) constitutes the only usable criterion for delimiting inventions against other kinds of intellectual achievements, and therefore technicality is a precondition for patentability (the "Logikverifikation" decision of the Federal Court of Justice (BGH) sees Art 27 TRIPs as "posterior confirmation" of this jurisprudence). The exclusion provision of Art 52 (2) and (3) EPC can also not be construed to be in conflict with Art 27 TRIPs, since it is based on the notion of lacking technical character of the excluded items."

The European Parliament specifically stated that:

“Member states shall ensure that data processing is not considered to be a field of technology in the sense of patent law, and that innovations in the field of data processing are not considered to be inventions in the sense of patent law.”

The Swedish Patent Court of Appeals noted that:

“As pointed out by the plaintiff, Sweden is bound to follow the rules in the TRIPs agreement since it joined the WTO in 1995. What's relevant to this case is that article 27 (1) says that the possibility to get a patent should be available for every invention in any technical field. This decision has not made any change in § 1 PL [Swedish Patent Law] necessary. Neither article 27 (1) nor any other part of the agreement gives a legal definition of the concept "invention".”

A EU commissioned study in 2002 went further in clarifying the patentability of software patents in TRIPS:

“Proponents of software patenting have argued that Article 27(1) does not allow software from being excluded from patentability, since computer software is to be considered a "field of technology". The discussions preceding adoption of the TRIPs agreement, however, do not confirm such a reading. In the absence of a legal definition of "invention", the agreement arguably leaves it to the member states to determine what constitutes a patentable invention, and whether or not that includes computer software as such.”

As such, it is clear that software patents are not automatically granted by TRIPS provisions and it is up to member states to determine the extent of patentability of software related inventions.

3. Harm Caused by Software Patents

The introduction of software patents has introduced a loud debate as there are strong concerns as to the harm caused by such patents by stifling innovation in the rapidly changing field of the software industry as well as documented cases of abuse by unscrupulous companies.

3.1 The Inherent Nature of the Software Industry

Software engineering, like other fields of engineering, require specialized knowledge and skills. There is, however, one big difference – software is always built out of ideal mathematical objects whereas other forms of engineering is dependent on some form of physical interaction. As such, building software is fundamentally easier compared to building hardware. Richard Stallman phrased this beautifully:

*“When we programmers put a **while** statement inside an **if** statement, we don't have to worry about whether the **while** statement will run such a long time that it will burn out the **if** statement, or that it will rub against the **if** statement and wear it out. We don't have to worry*

that it will vibrate at the wrong speed and the if statement will resonate and crack. We don't have to worry about physical replacement of the broken if statement. We don't have to worry about whether the if statement can deliver enough current to the while statement without a voltage drop. There are many problems of hardware design that we don't have to worry about."

As such, software developers can create vastly complex systems very quickly. Each of these systems contain hundreds, if not thousands, of algorithms and techniques, which are weaved together to build a copyrighted work. By the current standards of the patent office, all these algorithms can be patented. When these algorithms are patented, it becomes impossible to write software without infringing on patents. The cost of patenting individual algorithms would drive many software developers to the ground. It is therefore not surprising that 80% of software companies in Europe actively oppose software patents

The USPTO has also granted patents on combinations of algorithms and techniques that produce a particular feature. The best example in this case would be Apple Computers, who was sued because their revolutionary Hypercard program allegedly violated a patent which allowed for scrolling through a database and displaying selected parts of each line of text. This patent claim had no novelty as many software developers would be able to develop such a system independently. Patenting that particular idea resulted in no net intellectual value gain for the monopoly the patent owner was granted.

The case for more protection has not been made satisfactorily. There are no vendor neutral studies that make an effective case for software patents. This pinpoints the fundamental flaw in the software patent policy: that software inventions are rare and precious and therefore must be protected. All evidence points to the contrary.

The software industry has distinguished itself as the fastest growing industry ever, *without* having patent protection. The three decades of the personal computer software industry has seen multiple revolutions of change, and this rate of change is unprecedented and unique to the software industry. No other industry shares this characteristic. Technology development in the 20th century, particularly in the software industry, has been distinguished by the ability of small companies to take market leadership by developing new or innovative software to rival with their bigger counterparts. Copyright has always deemed to be sufficient for intellectual right protection. Indeed, as William Henry Gates III, founder of the largest software company, in the world stated in 1991,

"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."

In this vein, it is striking to note that had IBM patented its personal computer, Compaq would not be able to reverse engineer the IBM BIOS and the personal computer revolution would not have occurred in the same vibrant manner that it did in the 1970s and 1980s.

Finally, the search for prior art for software patents is almost impossible given the fast evolving nature of the industry as well as the relative youth of the industry. Given the limited resources of the patent office and the large number of patent requests as well as the large

body of software, it is difficult for patent examiners to properly examine prior art relating to the patent. This leads to overly broad patents being granted, to which litigation has proved necessary for the revocation of the patent.

There are many techniques that were never documented because they were considered obvious by the programmers. Hence, patent examiners cannot find proof of prior art among published literature. For example, AT&T sued MIT as MIT's X Window System used the concept of "backing store" (US patent claim number 4555775). This technique was developed by MIT before AT&T applied for a patent but the concept was never published by the programmers (except in a programmer's manual) so the patent examiners could not effectively check for prior art.

3.2 Hindrance to the Development of Free and Open Source Software (FOSS)

FOSS refers to computer software that users are free to run, copy, distribute, study, change and improve on. Anyone can take the programs, understand how they work, use them and redistribute copies without asking for permission. The "open" concept allows further customization of the software to users needs. The well-known Linux operating system is an example of FOSS software. Other examples include the OpenOffice.org productivity suite and the Firefox web browser.

Software patents inhibit the development and use of FOSS. The nature of FOSS is that many different bodies, agencies and companies help out for the development of software that would meet their respective needs. Nobody will be willing to pay licensing fees as required by patents and this will inhibit the development of FOSS.

There are other documented cases of harm:

- The Unisys patent of the LZW data compression algorithm. The LZW data compression method was published in an issue of the IEEE journal. The readers of the journal were not told that the authors of the data compression method had applied for a patent. Much public domain software, including the GIF image format, was created and used the LZW data compression method. Several years hence, Unisys notified its intention of enforcing its patent. This was done despite the fact that the data compression method was already widely used by companies and FOSS developers. FOSS developers had to remove all references that used the LZW data compression method.
- Microsoft has published a new license for its file sharing service (also known as CIFS). This license specifically forbids FOSS under GNU GPL, LGPL and similar licenses from not using CIFS. This is based on two broad and trivial US patents with priority dates of 1989 and 1993, and Microsoft's new terms effectively stop interoperability of its software with any other software.
- Adobe filed a lawsuit against Macromedia for infringing on its patent (5546528) which lays claim to the idea of adding a third dimension to menus by grouping them as tabbed palettes one behind the other. The damages awarded to Adobe was in the range of USD 2.8 million. The EPO has granted EP0689133 with exactly the same set of claims and priority date of 1994-06-23 on 2001-08-06, after five years of

examination period. Many programs, including free software such as The GIMP, infringe on this patent and will need to pay royalty fees if Adobe requires for it.

- Microsoft patented the ASF media file format and stopped a FOSS video capture and processing program for Windows from supporting their format. This has severe consequences as it hinders interoperability efforts.

Other cases of documented harm are available at [8]

4. Conclusion

Malaysia's software industry is nascent and to encourage its developers to branch out globally both in software products and services, we need to ensure that sufficient protection is available for commercialization of software.

The development of software involves the usage of ideas and functions which others have developed before. If these ideas are patented, before any piece of software can be written, a tremendous amount of resources have to be expended to check for patent infringement and after that the work needed to come up with alternative methods or to circumvent the potential patent violation areas will be huge in most cases. So, to play safe, a developer should then license the affected software patents or enter into cross licensing agreements with the entities which hold the patents. While larger software companies may be able to perform these as they will have the necessary resources and legal infrastructure in place, the same cannot be said for small startups. These additional burdens will be hefty for startups, many of whom will definitely lack the resources needed to tackle this issue. This will surely smother the nascent software industry we have today.

If the patents office lacks adequate technical and/or manpower resources, this will lead to the danger that patent applications are not scrutinized adequately for prior art and thus, ideas and functionalities which are too generic or abstract will be granted patents. As it stands today, it is already proving to be very difficult for patents offices in countries like US not to make mistakes like these, and examples abound of bad software patents being granted (and overturned, in some cases, when challenged in court). Again, it is the small startup which lacks the legal and financial resources to challenge any patent lawsuit which will suffer the most.

Currently software patents are not explicitly recognized in Malaysia and it should remain so. There is no hope of us ever achieving technological independence and building our domestic software capacity if software patents are allowed. Local software developers and companies will suffer as they have little or no patents arsenal to fall back on in the case of a software patent infringement lawsuit and many will also lack the financial and legal resources needed to fight a lawsuit. It will surely blow out whatever spark there is currently in our fledgling local software industry and in the end only large established multinational software companies will be left here.

5. References

[1]

<http://216.239.35.100/search?q=cache:liEo88nkcL0C:www.slwk.com/av/asia.pdf+%22software+patent%22++malaysia&hl=en&ie=UTF-8&e=980>

[2] <http://www.kpdnhq.gov.my/ip/patent1.html>

[3] <http://www.lawyerment.com.my/intellectual/patent.shtml>

[4] http://www.iplcca.com/myip/res_patent.html

[5] http://www.iplcca.com/faq/pat_faq.html

[6] <http://www.ladasperry.com/GUIDES/COMPUTER/Patents.USA.html#PatIntro>

[7] <http://www.ladas.com/GUIDES/COMPUTER/Computer.EPOJP.html>

[8] <http://swpat.ffii.org/patents/effects/>

[9] <http://swpat.ffii.org/analysis/trips/index.en.html>