Software Useright: Solving Inconsistencies of Software Patents

Version 0.9

September 15, 1999

Abstract

This paper gives an overview of the principles, the economic impact and the potential juridical contradictions of patents and especially software related patents. It is structured in five parts, namely: History, Law, Economy, Justice and Solutions. The first part (History) shows that the historical legitimity of patents is to "encourage people to share technology". The second part (Law) gives an overview of what is considered to be an invention and what is not. The third part (Economy) introduces a model to determine which patents are useful and which are not. The fourth part (Justice) analyses which patents are fair and which are not from a citizen point of view. The fifth part (Solutions) introduces the concept of Software Useright, an integrated set of legal approaches which make software related patents useful and fair. Well implemented patent law based on Software Useright may even lead to an increase of safety and consumer protection in the software industry.

Definitions

Computer Programme A computer programme is a serie of instructions in a computer language which implements an abstract specification.

Theorem (Curry-Howard): computer programmes are equivalent to the mathematical proof of their abstract specification.

Software A software is made of

1. Images
2. Sounds
3. Texts
4. Computer programmes

Software can be protected by copyright law. In some countries (USA, Japan), computer programmes are considered as inventions and can be patented.

1 History: why were patents introduced?

1.1 A short story

Imagine a workshop in the middle age, which is making the best swords in the kingdom thanks to some secret process, known only by the operatorof the workshop. Then comes war, or disease, and the old operator dies, together with his secret invention. The best swords in the world will never be produced again. As a result, all remaining human beings will never be able to benefit from some very valuable invention.

Let's imagine now another history: one day, one of the workers at the workshop looks through a hole in the wall and discovers the secret process used by the operator of the workshop. He then tells the story to his friends. One of them creates his own workshop to manufacture similar swords, in terms of technology, but with a better industrial design. As a result, customers start buying less and less swords from the orginal workshop which, in the end, has to close down because it could never be proved that the new workshop copied the original process.

Patents were created to give some kind of solution to the two versions of this short story. Firstly, a patent is granted for some kind of valuable invention if, and only if, this invention is clearly explained and published. This solves the issue of valuable inventions that can be lost if they are kept secret. Secondly, the patent owner is granted a monopoly on the use the invention. This gives him a much more efficient way to protect his interests against people who copy his invention than trying to keep it secret.

1.2 Patent law

Patents are granted on inventions. We will see in the next section what can be considered as an invention. Monopolies on inventions are granted for a certain amount of time. Nowadays, the TRIPS[1] agreements signed by all members of the World Trade Organisation state that patent monopolies should last 20 years, whatever the kind of technology they are granted for and whatever its value.

Patents are granted on a national base, just like trademarks. A patent granted in Europe cannot be enforced in the USA. A patented granted in the USA cannot be enforced in Japan, etc. In order to get a worldwide monopoly, one must file for a patent in each country.

The case of Europe is somehow complex. The Munich European Patent Convention[2] (EPC), signed by 19 European states, gave birth to a European Patent Office[3] (EPO) located in Munich, Germany. At the same time, most member states of the EPC have their own national patent office and their own national patent law. The agreements between the EPO and the national patent offices together with the inconsistencies between the EPO jurisprudence and

[1] TRIPS agreements - http://www.wto.org/wto/intellec/1-ipcon.htm
[2] http://www.european-patent-office.org/epc97
[3] http://www.european-patent-office.org

the national laws result in a situation where some patents can be granted by a national patent office and rejected by the EPO and some others can be granted by the EPO but cannot be enforced in some member states of the EPC.

Two draft directives are currently being prepared by the European Commission (DG XV) in order to simplify the implementation of patents in Europe. The first draft directive is related to patents[4] and has not been published yet. The second directive is related to utility certificates [5] and has been published on the European Commission web site. Utility certificates are very similar to patents except that the duration of the granted monopoly is shorter (6 to 10 years only), that a utility certificate is cheaper to be granted and that no review or prior art search is necessary to be granted this certificate.

1.3 Patent infringement

Someone who provides a patented invention without the permission of the patent owner can be sued for patent infringement. In most countries, it does not matter whether the infrigement was made knowingly or not. This means that someone who starts using a patented invention, and does not know that the invention had been patented, can be sued by the patent owner for patent infringement. If he looses, he is liklely to be ordered by the judge to pay to the patent owner an amount of money equivalent or bigger to the sales that were lost because of the existence of an infringing competitor. This is very similar to what happens when an "unfair competition" situation can be proven.

Sometimes, a patented invention is made of various subparts. Instead of buying the invention, one could buy the various subparts which constitute the invention. Let's see what the French law[6] has to say. In this case, the user can be sued for patent infringement only if he is aware that he is knowingly infringing a patent. And the company providing the subparts of the invention can only be sued if it is aware it is knowlingly providing the necessary subparts to assemble a patented invention (L613-4 of the French Intellectual Property Code). In practice, sending a warning letter is therefore necessary before starting suing someone for knowingly providing subparts of a patented invention. Therefore, the risk of being sued for providing subparts of patented invention without being aware of it is very low.

1.4 Software is seldom secret

Most packaged software are published. And, thanks to the use of decompilation, a technique which consists in automatically reverse engineering the binary code in order to produce human readable source code, it is quite feasable, although rather long, to discover the internals of a complex piece of software. Thus,

[4] http://europa.eu.int/comm/dg15/en/intprop/indprop/99.htm

[5] http://europa.eu.int/comm/dg15/en/intprop/indprop/utility.htm

[6] French law is a typical continental law which makes a theoretical discussion on abstract principles fairly easy. Most codes of the French law are available on the web at http://www.legifrance.gouv.fr

from a technical point of view, the argument that patents incite people to share secret technology does not apply to packaged software since technology is automatically published and shared as soon as software is published. Of course, decompilation and reverse engineering take time: it would probably take one year to three years before a specific feature or technology can be understood and copied into another software. Let's consider for example the time it took for word processor makers to reverse engineer Microsoft's Office 97 file formats and make their product fully compatible with Microsoft products: two years.

There are exceptions to this general situation because all software are not published. Here are some examples of non published, still potentially inventive, software:

1. Software engineering tools such as compilers, code optimizers, rapid application development libraries may be used by a company internally to produce packaged software without being published themselves,
2. Software running on web servers may provide a public service on the Internet without being published themselves,
3. Custom software used internally in a corporation to monitor an advanced industrial process.

One may argue that such exceptions justify the use of software patents in order make sure that any software related invention is going to be published and shared, especially considering the increasing number of web services based on secret custom software.

1.5 Patents and decompilation laws: potential contradictions

Decompilation is not legal in many countries. It is illegal in the USA. It is illegal in Europe except for interoperability purpose. It is legal in Japan except for USA made software[7]. It became legal in Australia lately[8], but only for interoperability and error correction purpose. Thus, most copyright licenses on software include legal clauses which forbid the use of decompilation.

This situation creates some difficulties in order to prove a patent infringement. Let's imagine the following situation. A company called "nicecompany" develops and markets a new 3D software which introduces a real-time implementation of the ray-tracing algorithm thanks to a new patented technology. Another company, called "metoocompany", decompiles, in a country where it is legal, the new software from "nicecompany" and asks a couple of highly-skilled engineers to reimplement the patented technology in their own software. After one year "metoocompany" releases a new 3D software which renders scenes 50% faster than the software from "nicecompany" under a copyright license which

[7]The USA was able to force Japan to accept a bilateral convention on decompilation which only applies to USA made software. This is typical of Japan USA relations.

[8]http://www.dcita.gov.au/cgi-bin/trap.pl?path=4189

prohibits decompilation. At this point, "nicecompany" believes that "metoocompany" decompiled their software and copied the patented technology. They may decide to sue "metoocompany" for patent infringement. But how are they going to make evidences. They need to decompile the software of "metoocompany" in order to exhibit the proof. And they may get sued by "metoocompany" for doing this in a country where it is illegal.

This story is not as theoretical as it may seem. For example[9], in 1994, company Stac won against Microsoft for patent infringement. However, Microsoft won against Stac for trade-secret misappropriation because Stac used decompilation on the software made by Microsoft. Stac had to pay Microsoft 13.6 million USD in dammages to Microsoft and Microsoft had to pay 120 million USD in dammages to Stac.

The contradiction goes even further: let us imagine that "nicecompany" publishes their software with its source code in order to provide better service to its customers. Because the software contains thousands of elementary processes and because many of them are patented, it is almost certain that "nicecompany" is infringing on someonelse's patent without being aware of it. Because "nicecompany" publishes their software together with its source code, it is very easy to show evidences of patent infringement. Anyone, including "metoocompany", could sue without risk "nicecompany" for infringement on some obscure patent that nobody in the 3D industry was aware of.

As a result, few companies are going to publish the source code of their software which is in contradiction with the historical goal of patents: "encourage people to share technology".

Of course, if the patent infringement is related to user interface techniques, decompilation is not necessary to show evidences. On the other hand, user interface techniques are very seldom kept secret, which means that there are no justications for patents on user interface from a historical point of view. Moreover, patents on user interface raise specific concerns related to interoperability and standardization as we shall see bellow.

2 Law: what is an invention?

Patents were introduce to encourage people to publish their inventions and share technology. But what is an invention? In Europe, an invention must be "new, inventive and susceptible of industrial application"[10] while in the USA it only need to be "new, non obvious and useful"[11].

What about software? We will see that there two kinds of patents on software: patents on inventions based on innovative software and patents on programmes as such.

[9] "LA Law". Andrew Schulmm. Document found by Roberto Dicosmo, published on the web site of CSIRO (Australia) and unexpectedly removed after the publication of M. Dicosmo's essay on Microsoft.

[10] Article 52 of the Munich convention.

[11]

2.1 New, inventive, industrial application

Article 52 of the Munich convention states that "European patents shall be granted for any inventions which are susceptible of industrial application, which are new and which involve an inventive step" and adds in paragraph 2 that the the following in particular shall not be regarded as inventions:

1. discoveries, scientific theories and mathematical methods;
2. aesthetic creations;
3. schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;
4. presentations of information.

However, paragraph 3 of article 52 states that the provisions of paragraph 2 only apply to the subject-matter or the activities as such. In particular, an invention which satisfies the novelty and industrial application criterion and which inventive step is related to programmes for computers can be patented. That's how 13.000 patents have been granted in Europe on software related techniques.

Article 54 states that "An invention shall be considered to be new if it does not form part of the state of the art" and that "the state of the art shall be held to comprise everything made available to the public by means of a written or oral description, by use, or in any other way, before the date of filing of the European patent application."

Article 56 states that "an invention shall be considered as involving an inventive step if, having regard to the state of the art, it is not obvious to a person skilled in the art."

Article 57 states that "An invention shall be considered as susceptible of industrial application if it can be made or used in any kind of industry, including agriculture."

Article 57 may be subject to interpretation from one country to another because the word "industry" has different meanings in different countries. For example, one may talk about the "banking industry" in English while it may sound strange to talk about the "*l'industrie de la banque*" in French since banking is considered as a service activity rather than as an industrial activity in France.

2.2 New, technical solution of a technical problem, industrial application

Recently, the European Commission as well as various European governments have issued public documents[12] which may contradict the legal definition of inventions. For example, the French government has sent a letter to a citizen

[12] Some copies of these documents are available on http://www.freepatents.org/

where it is written that "inventions are considered by the EPO as technical solutions of technical problems". Such definition of inventions may possibly allow to patent many more things because the "industrial application" criterion has been removed from the definition. It is also much closer to the USA definition of inventions.

Still, the legal definition of inventions in Europe requires inventions to be "new, inventive and susceptible of industrial application". And the approach known as "technical solution of a technical problem" has actually no direct relation with the legal definition of inventions in articles of the law. It only has a relation with the application rules of the Munich convention. Rule 27 of the EPC defines how inventions should be described in the patent application form and states that one "should disclose the invention, as claimed, in such terms that the technical problem (even if not expressly stated as such) and its solution can be understood".

Rule 27 can actually be understood as a particular definition of inventivity rather than as a general definition of inventions. Even if an EPO court decided in a case that some "new technical solution of a technical problem with no industrial application" could be considered as an invention, the patent granted in such case by the EPO could be easily cancelled since judges in Europe (besides the UK) may not contradict articles of the law. It is therefore misleading to define in official European documents inventions as "technical solutions of technical problems". It would be much more appropriate to define inventions in Europe as "technical solutions of technical problems, which are new and susceptible of industrial application".

2.3 New, non obvious, useful

In the USA, inventions need only to be "new, non obvious and useful" which allows to get patents for about anything including many activities and goods with no industrial application. Business methods, consulting methods, mathematical formulas can be patented as long as they have "a useful, concrete and tangible result" said the United States Court of Appeals for the Federal Circuit, which handles all patent appeals[13].

2.4 Patents on inventions involving innovative software

There are two kinds of software patents:

1. patents on inventions which include some software process,
2. patents on computer programmes as such.

This distinction may sound futile in the USA where both are legal. It is very important in Europe where a computer programme as such is not considered as an invention.

[13] http://www.nytimes.com/library/tech/99/01/biztech/articles/12patent.html

On the other hand, patents on inventions involving innovative software are legal in Europe as soon as they satisfy the three conditions: "novelty, inventivity and industrial application". For example, there are patents in Europe on oil exploration processes based on innovative software. Although the innovative part of the process resides solely within the software, the whole industrial process can be considered as innovative and is of course susceptible of industrial application. The same arguments apply for a process, such as the MP3 technology, which includes some software and allows to decompress digitized music and listen to it. The fact that the process has some impact on sound, allows to say that it has an industrial application.

The distinction between inventive and non-inventive, between susceptible of industrial application and not susceptible of industrial application in the case of software largely depends on the European Patent Office jurisprudence[14]. Patents on oil exploration processes involving software were for example subject to debate 10 years ago. Nowadays, patents on user interface techniques are getting more and more accepted by the European Patent Office[15] without any changes to the Munich convention.

2.5 Patents on computer programmes as such

Currently, computer programmes as such cannot be considered as inventions in Europe. This has two important consequences:

1. A software publisher cannot be sued for patent infringement,
2. Software related inventions with no industrial application cannot be patented.

The current situation in Europe is very favorable to software publishers. As we saw before, only a whole invention involving innovative software can be patented. Thus, only a whole process involving innovative software can be considered as infringing on a patent. Software itself is only a sub-part of the whole patented process. Therefore, software publishers can only be sued for providing some sub-part which enables their customers to assemble and use a patented process. This can only be done by notifying them first that they are providing some sub-part of a larger patented process whereas suing someone for direct infringement can be done without notice. It gives some kind of protection to software publishers against "legal terrorism". It is also good for businesses which distribute CD-ROMs including thousand of third-party software components since there is no risk to be sued for infringement. This is for example very important for Linux distribution[16] publishers such as SuSE[17] or Mandrake[18].

[14] "La brevetabilite des inventions". Frederic Polland-Dulian, IRPI, ed. Litec 1997.

[15] http://www.european-patent-office.org/dg3/biblio/t970935eu1.htm

[16] A Linux distribution such as SuSE contains more than 1300 software, each if which countains thousands of elementary processes that could be potentially patented. If programmes as such could be patented, this would create a risk of being sued for about any already patented process related to software.

[17] http://www.suse.de

[18] http://www.linux-mandrake.com

The current situation in Europe also keeps out of the way patents related to business methods or social practices implemented through computer programmes. The situation experienced in the USA where any programme which implements an existing social practice can be considered as "new, non obvious and useful" cannot happen in Europe with the current law.

The fact that some new and inventive process is implemented through a computer programme is not sufficient in Europe for such or process to be considered as satisfying the "industrial application" criterion. However, there would be a risk, if computer programmes as such were considered as inventions, that such new and inventive processes not susceptible of industrial application would be considered as inventions as soon as they are implemented through a computer programme. The same kind of risk would also exist if the "industrial application" criterion was removed from the general definition of inventions: many consulting activities have no industrial application but are still technical.

Of course, if computer programmes as such could be patented in Europe, software publishers could be sued for patent infringement as soon as they distribute or copy, knowingly or not, an original software involving techniques patented by others. And if new "technical solutions of technical problems" not susceptible of industrial application were considered as inventions, many computer service companies could be sued for performing, knowingly or not, patented service and consulting processes.

2.6 Software patents and interoperability principle: potential contradictions

The European copyright law for software includes various interoperability principles, in particular regarding decompilation[19]. On the other hand, patents granted on software related inventions may entitle the patent owner to forbid anyone else to use a given patented technique in its software. For example, patents on standard music compression processes may be used to forbid the development of third party compatible software.

This situation is already happening in Europe with the MP3 *de facto* standard. The French company Thomson has already threatened to sue a young swedish programmer because he was developing a software compatible with the MP3 standard[20]. It seems that clarifications are needed in order to make patents on inventions involving software compatible with the interoperability principles of the European copyright law for software.

Moreover, patents on user interface techniques tend to have the same effects on competition as patents on technical interfaces. If users get used to a certain kind of interface, it gets very hard for them to switch to another kind of interface. Imagine for example if some cars had steerwheels and others joysticks because steerwheels and joystick were patented and licensed under a exclusive contract to a restricted set of car makers. Because the European Union was designed

[19] http://europa.eu.int/eur-lex/en/lif/dat/1991/en_391L0250.html

[20] I can not find this reference. I read his site forgot to bookmark it. Help needed.

with the idea that free competition in a free market should be regarded as very important, directives related to software patents should take into consideration the potential damages of user interface patents on free competition.

Various approaches may be implemented in order to garantee free competition in a free market with software patents. In particular:

1. Non exclusive and fair licensing policy for interfaces,
2. Free licensing policy for interfaces,
3. No patents on interfaces.

The first approach is the most complex: it requires a regulation authority responsible for evaluating licensing policies of patent owners and enforcing the non exclusive and free licensing policy for interfaces. It should also be consistent with the European software law (copyright) and should not discriminate one industrial model against another. In particular, patent law should not restrict the right for authors to grant the right to users or distributors to copy a software. Otherwise, it would generate a high risk for authors or distributors, and would create in particular a hostile environment for free software. The current situation with MPEG2 video technology, where owners of the standard are using their patent rights and try to eliminate MPEG2 compatible free software is the typical kind of situation that should not be tolerated. The fifth part of this article introduces a solution which allows to protect the interest of patent owners without discriminating one industrial model against another.

The second approach is probably much easier to implement. Patents could be granted on interfacing techniques with industrial application. Still, anyone could use such interfacing techniques for free. There is no need for a regulation authority.

The third approach is probably very hard to implement in terms of law because some interfacing techniques are new and susceptible of industrial application. Rejecting patents on such interfacing techniques would constitute an exception.

2.7 Computer programmes and mathematics: potential contradictions

It was demonstrated that any computer programme is scientifically equivalent to a mathematical proof while article 52.2 prevents mathematical methods to be patented as such. Removing computer programmes from the list of exceptions of article 52 would lead to a situation where the same object would be at the same time patentable and non patentable under article 52.2.

The current draft directive on utility certificates does not state that computer programmes as such are not inventions. This is both in contradiction with the Munich convention and with the fact that programmes as such are equivalent to mathematical proofs.

2.8 Patents & Free Software: potential contradictions

Free Software research & development involves publishing the software on the Internet or elsewhere. If computer programmes as such would become patentable inventions, are we going to consider that:

1. Free Software belongs to R&D and that patents do not apply to them?
2. Free Software development is a non-commercial activity held by a private community?
3. Free Software is a commercial good and therefore can be considered as potentially on patents?

As a reminder, here is an excerpt of the French patent law (L613-5 of the French Intellectual Property Code): "Les droits conférés par le brevet ne s'étendent pas : a) aux actes accomplis dans un cadre privé et à des fins non commerciales ; b) aux actes accomplis à titre expérimental qui portent sur l'objet de l'invention brevetée ..." which means that the rights granted by a patent cannot be enforced on private and non-commercial activities as well as on experimental activities based on the patented invention.

Of course, when programmes as such are not patentable as it is currentlty the case, patents on software related inventions cannot be enforced for "non commercial activities in a private situation". In the current situation of national laws and article 52.2 of the Munich convention:

1. People are free to use Free Software involving patented techniques for private & non-commercial purpose,
2. People are free to provide Free Software involving patented techniques for private & non-commercial use,
3. People are free to develop Free Software involving patented techniques.

However, they are not free to use Free Software for non private or commercial purpose. A patent license for all patented software techniques is required for any non private or commercial activity.

2.9 Sofwtare patents & Article 100: potential contradictions

Article 100A of the Rome Treaty[21] is the main justification for the European Commission (DGXV) to propose a harmonisation directive on patents in Europe. The application to software patents and their legalisation are only a sideproduct of this larger effort.

Paragraph 3 of article 100A states that "The Commission, in its proposals ... concerning health, safety, environmental protection and consumer protection,

[21] http://europa.eu.int/abc/obj/treaties/en/entr6d05.htm

will take as a base a high level of protection". As we can see with the Y2K bug, software is crucial for security in the modern world. And software published with its source code is much easier to correct than software published without source code, especially when the software publisher has disappeared. In the space industry or the nuclear industry, the availability of the source is considered as compulsory in order to reach a high level of security. Secret source code for a strategic software component such as a compiler may lead to severe bugs such as the one which lead to the failure of the Ariane V rocket.

The problem with software patents, and more specificaly patents on programmes as such, as we could see in the first part of this paper, is that they may encourage people to stop publishing the source code because it is harder, if not illegal, to prove a patent infringement on binary code, especially if decompilation is illegal. Software patents may actually reduce security in the software industry. Thus, there is a contradiction to Article 100A of the Rome Treaty in such a way that a high level of safety is not garanteed. Solutions to this problem will be suggested in the last part of this paper.

On the other hand, a situation where consumers become responsible for patent infringement cannot be considered as an increase in termes of consumer protection. We will also see in the last part of this paper that simple measures, which do not require changes in the law, can be suggested in order to protect consumers from undesired infringement.

2.10 Generalisation, equivalence, meta-programming and adaptivity: potential inconsistencies

Computer science is often a matter of discovering and naming new concepts which are more general than former ones. And many new technologies are just the generalisation of particular applications of existing technologies. For example, many people actually did some kind of object-oriented programming before object technology was officially named and invented. But it was not named "object technology" and was not as general as "object languages" are. What would have happened if some particular object-oriented technique had been patented before the introduction of more general object languages? Would general object languages infringe on a particular object-oriented technology?

Also, there are many equivalences in computer science. Again, object technology shares many similarities with functional programming but the names related to each technology are quite different although the concepts are equivalents. Would a patent on object technology apply to functional programming and vice-versa?

Finally, more and more computer programmes are not written by humans but rather by computers either under the control of humans or autonomously. This is called meta-programming. Some meta-programming techniques, based on evolution and simulation, produce computer programmes with unsuspected and very inventive techniques. How can such inventive techniques be patented? Who do they belong to in the case where the program generator behaves autonomously?

Some meta-programming techniques even involve computer programmes which modify themselves in order to adapt to their environment, without any human control. If such a programme modifies itself and then implements a patented technique, who is responsible for the infringement?

Because adaptive and meta-programming are very important approaches in the modern computer industry, patent law should not discriminate them against more traditionnal programming approaches by creating a legal risk. Banning software patents on techniques generated by autonomous computer programmes is probably the most reasonable approach in a first step.

3 Economy: are patents useful?

The main economic justification of patents is to finance innovation. Which private investor would invest money in research and development if there wasn't a way to get some kind of reward on the results for example through a patent which could be sold as an exclusive property title. However, the software market has grown very quickly without patents. Companies like Microsoft could dominate the market without ever filing or using a patent. And many innovations come from profitable small companies that never file patents.

3.1 Elementary patent theory

We consider patents in afree market with a perfect knowledge of the value of inventions.

3.1.1 Actors

We distinguish four elementary economic actors: patent producers, patent traders, patent licensees and consumers.

Patent producers Patent producers invest money in order to find inventions for which they can get a patent. In the real world, patent producers are the combination of researchers, patent experts and lawyers.

Patent traders Patent traders buy patents from patent producers or other patent traders in order to sell licenses to patent licensees. Licenses can be exclusive or non-exclusive. The main role of patent traders is to mutualise the development costs of patents among patent licensees and to optimise the use of a patent by selling it to the economic actors who can pay the most for its use.

Patent licensees Patent licensees buy licenses from patent traders in order to be allowed to use patented inventions in the products they sell to consumers.

Consumers Consumers buy goods containg inventions from patent licensees.

Most private companies act both as patent producers, patent traders and patent licensees. However, this makes analysis very complex since it is hard to decide in such situation where the profit or the losses come from. Moreover, the modern economy sees an increase of specialised actors such as private research companies, independent patent funds, etc. Therefore, a clear separation of producers, traders and licensees, allows to have a single model for both integrated companies and specialised actors.

3.1.2 Economy

One of the usual justifications of patents is to finance innovation and the other is to reward the innovator for the investment in sseking the innovation. Let's see how.

Patent producers The patent producer invests $D = D_1 + D_2$ to produce a new patent. The production costs of a patent include development costs D_1 (researchers) as well as administrative costs D_2 (patent experts, lawyers, patent office, etc.). The patent brings to the end consumer a benefit of b.

The patent producer receives P from the patent trader when it sells the patent.

Patent traders The patent trader buys a patent at price P from a patent producer. He sells a license of the patent to l patent licensees at price $L_1..L_l$. If $l = 1$, or if only a restricted community can get a license, the license is said to be exclusive. If anyone can get a license, the license is said to be non-exclusive.

Patent traders may also buy and sell patents to other patent traders. We shall not cover this issue here.

Patent licensees A patent licensee buys a patent license at price L from a patent trader. He invests I to industrialise the patent and sells goods to consumers at price g.

Consumers Consumers buy n goods from the patent licensee at price g and spend $G = n.g$. The utility u provided to consumers will be considered as equal to $b - g$, that is what they benefit from the good minus what it costs to them. The global benefit $B = n.b$ and the global utility $U = n.u$ will be used hereafter.

3.1.3 Simple case

We suppose that patent traders only sell exclusive licenses. In order to make the whole economy profitable for everyone, patent producers must cover their expenses P when they sell a patent:

$$P > D$$

Patent traders must sell patent licenses at a higher price that what they spent to buy the patent:

$$L > P$$

Patent producers must sell enough goods to consumers to cover both the licensing and industrialisation costs:

$$G > L + I$$

And consumers will buy goods if the utility is positive:

$$U = B - G > 0$$

This brings the following global relation:

$$B > G > L + I > P + I > D + I$$

The first meaning of this relation is that the price of goods sold to consumers should be high enough to cover the development costs D of the patent as well as the industrialisation costs I. This is why patents are supposed to be good to finance innovation since only patent licensees are allowed to industrialise innovation and thus must contribute to it by funding it.

Another meaning of this relation is that the price of goods sold to consumers should never be higher than the benefits B provided to consumers. Because some goods can be very expensive to industrialise, and because the market or the benefits provided to consumers may be rather small. It may for example happen that the global benefits provided to consumers would never cover the global industrialisation costs of two patent licensees working independently ($B < 2.I$). This situation justifies the concept of exclusive patent in order to prevent two patent licensees to compete with the same patent with no chance of being profitable.

Finally, if we consider this relation from a consumer point of view, we get:

$$b > g > \frac{L+I}{n} > \frac{D+I}{n}$$

It shows that, if the benefit b is rather small compared to development costs, only patent licensees which can sell goods to enough consumers will be able to cover those costs. This also justifies granting patents for a long period (ex. 20 years) in order to give a chance to new entrants to find customers and sell enough goods.

3.1.4 Introducing risk

Not all research are successful. The inherent risk of innovation can be represented by saying that only one patent development out of α will be sold to patent traders. If we consider the global economy of innovation, we get something like:

$$B > G > L + I > P + I > \alpha.D + I$$

We then get from a consumer point of view:

$$b > g > \frac{L+I}{n} > \frac{\alpha.D+I}{n}$$

The difference with the previous situation is that the price of goods sold to consumers should be high enough to cover the development costs $\alpha.D$ of both successful developments and unsuccessful developments.

Considered from a macro-economic point of view, this equation is the fundamental equation used to justifiy that research is a private activity that can be financed by the market and does not need public funding. However, it requires patent traders to buy patents at a much higher price than the development costs. This situation may only happen in a fair and competitive market.

3.1.5 Market evolution

Let's consider again the consumer point of view relation:

$$b > g > \frac{L+I}{n} > \frac{\alpha.D+I}{n}$$

It shows that companies with an existing installed customer base will be able to make profit from patents much more easily than new entrants without customer base. However, we may suppose that the market evolves in favour of new entrants whenever they provide better products.

We could suppose for example that two companies on the same market with n customers are competing and that the market share of each company evolves at a speed proportional to the differenciation of one product against the other. Ifτ is a typical latency coefficient of the market, the derivate $n_1^{'}$ of the market share of the first company is then computed through:

$$\frac{n_1^{'}}{n} = \frac{1}{\tau}.\frac{u_1-u_2}{u_1+u_2}$$

If the first company starts with no customers and a better product, it will be able to gain 100% of the market after δ years where δ is given by:

$$\delta = \tau.\frac{u_1+u_2}{u_1-u_2}$$

If the products are very similar with same price, it will take a lot of time for new entrants to take all the market and cover the development expenses. This shows that patents on minor or trivial inventions can only benefit to old & big players .

But if the product of the first company is much better than the other, it will take about τ years for the product of the new entrant to take all the market which is about the typical latency of the market. If the market is large enough (n important), a value of at least τ should be regarded as a fair approximation of what the duration of patent monopolies should be in order to give new entrants a chance to get in. If the market is small (n small), larger values of τ should be considered in order to garantee new entrants to cover the development costs of the patent.

3.1.6 Patents on legitimate inventions

An invention is considered to be legitimate if the following conditions are satisfied:

1. It requires much money to discover (D_1 large)
2. It is very useful (b large)
3. Administrative costs D_2 are non significant.

In this situation, equations can be simplified as

$$n.b > n.g > \alpha.D_1 + I$$

We are in the typical case where patents finance the research and development of inventions in a fair way.

3.1.7 Patents on minor inventions

An invention can be costly to develop (D_1 large) and not very useful (b small). In this case, only existing companies with a large installed customer base will be able to cover the development costs D_1. Patents still finance innovation but give a lead to big players in the market. This is a typical case in the consumer electronics industry.

3.1.8 Patents on trivial inventions

An invention can be trivial (D_1 small, I small). In this case, the equation can be simplified as:

$$n.b > n.g > \alpha.D_2$$

We are in a new situation where patents mainly finance the administrative costs of patents (lawyers, patent experts, patent offices, etc.) rather than innovation made by researchers.

Moreover, if the invention is not so useful (b small), we are in a situation where only the big players will be able to finance such costs.

3.1.9 Price setting of licenses

What is the market price L for a patent license? On the one hand, it should be cheaper than the global benefits B it provides to the consumers. It should also be cheaper than the benefits $n_i.b$ it provides to the installed customer base of each competitor. On the other hand, there is no reason for a dominant player of the market to pay more for a patent license than its immediate competitor.

In a competitive market with players of nearly similar size, patents will be licensed at a fraction of the global benefit they provide to customers. This fraction will be no bigger than the share of the second biggest company in the market.

In a situation with a dominant player and little competition, patents are going to be licensed at a low price, closer to $\alpha.D$ than to B, whatever the benefit they provide. In a quasi-monopoly situation, it is even likely that many patents will be licensed at a price close to D if no one else can propose a higher price. Since patent traders actually only need to buy the patents they are interested in, someone else, probably governments, will have to fund unsuccessful developments.

This shows that patents can only finance innovation in a really competitive market but not whenever there is a quasi-monopoly. In non competitive markets, patents will reinforce monopolies. In this case, governments will have to fund public research at a higher level, thus increasing public deficits.

In a situation without patents, governments spend $\alpha.D_1$. In a situation with patents, governments spend an additional $\alpha.D_2$ to file patents and receive P for the patents that can be sold to patent traders. Introducing patents is only directly profitable in terms of public deficit whenever $P > \alpha.D_2$. This situation does not happen currently for public research institutes such as INRIA (French

National Computer Science Research Institute). It is unlikely it will ever happen in a quasi-monopolistic market.

Moreover, introducing patents and reinforcing monopolies may also generate indirect costs:

1. Private national R&D on strategic software may disappear and will have to be replaced by more public R&D on strategic software,

2. Software price may be higher than in a more competitive market, thus requiring the governement to spend more taxpayers' money for its own information system.

3.2 Complex systems

Many goods like software, electronic components, etc. gather thousands of inventions in a single object and are called "complex systems" in this paper.

The problem with complex systems is that they can only be sold as a whole. It is therefore necessary to get a license for each invention in a complex system. Most complex systems sold in the market include infringements, not on purpose, but because it is very hard for complex system makers to know precisely which inventions they are using. As a result, complex system makers tend to fight constantly each other for patent infringement.

Whenever all actors have similar size, like in the microelectronics industry, legal battles end up with cross-licensing patents. Cross licensing means that one player allows the other to use its patents and vice-versa. In our model, this can be understood as forcing patent traders to grant exclusive licenses to some of the other players.

However, this makes life very hard for small players and new entrants. A new entrant may for example decide to get an exclusive license on a legitimate invention[22]. But, in order to build a complex system, he needs licenses on thousands of other inventions owned or licensed exclusively to old players. The only way to get those licenses would be to cross-license the legitimate invention with old players.

3.2.1 Cross-licensing and price-setting

What are going to be the price for this trade? On the one hand, the new entrant or the small player must accept the old & big players' conditions in order to manufacture a complex system. On the other hand, big players may make more profit if they could license the invention of the small player of the new entrant because the invention allows to sell their products at a higher price or make them more attractive. The potential profit that big players could make from licensing the invention is given by:

$$R = G - L - I$$

[22] As we saw above, the only way for a small player to get a license on a legitimate invention is to develop the invention himself. Otherwise, big players can always pay a higher price to the inventor and get an exclusive license on it.

In a cross licensing deal, big players could trade R against a small amount of market share for the small player with a value of at most R in terms of lost sales for them. Big players should therefore grant licenses at a price which allows small players to survive without being fierce competitors.

It is likely that small players will be granted a right to license inventions "owned" by big players at a price between g and b if, in return, they allow big players to use the small player's invention for free.

3.3 Software without patents

Software has experienced for 20 years tremendous growth and innovation without patents. The main reason to explain this is that software publishers who do not provide and sell better or upgraded versions of their software to their customers will not survive since software, unlike cars, never breaks[23].

3.3.1 Economy

The key economic actor is the software publishing company. It develops software, which is similar to industrialisation in the case of software with patents. It has to innovate because software development mainly consists in finding technical solutions to technical problems. The software publisher therefore also does the same kind of job as the patent researcher. Industrialisation and innovation are very closely linked through a single activity: writing software code.

Software publisher Software publishers invest D to find new features or techniques with a benefit of b to end consumers. They invest I to write software and sell software to consumers at price g. New features only become available to competitors after a certain amount of time τ_1 which depends on how easy it is to discover the features or techniques in the source code or in the binary code of the software once it has been published.

3.3.2 Simple case

The benefit provided to consumers must cover feature finding costs and development costs.

$$n.b > n.g > D + I$$

Usually, development costs are much higher than discovering new features or techniques. There are many trivial inventions with high utility in the software industry. Therefore, the equation becomes

$$n.b > n.g > I$$

[23] Of course, certain companies tend to develop software which becomes quickly obsolete or contains so many bugs that it has to be replaced every other year or month. The publication in 1997 (Office 97) or 1998 (Windows 98)of software products that were not compatible with the year 2000 bug is typical of such behaviour. But it should rather be considered, from a technical point of view, as an exception due to the lack of competition in the market. Competing products such as Linux show that most well designed software do not get obsolete quickly.

If the market if fluid enough ($\tau_1 > \tau$ where τ is the typical latency of the market), any invention that can be sold at once (within τ_1 years) to a sufficient number of customers can be financed. Such inventions can actually be self-financed without requiring venture capitalists. However, some inventions are expensive to develop and may require more complex business plans in order to cover development costs. Typical business plans (ex. machine translation software) start first selling the invention to a few happy rich customers and sell it after 10 years to the mass market. The rich customers may not be rich enough to cover all the development expenses of the invention. This requires retaining "ownership" on the invention 10 years in order to prohibit anyone to copy the invention in the meantime and garantee high prices when the invention if ready for the mass market.

Such inventions are hopefully very few in the software industry.

Something quite specific to software is that investment costs I are quite low. Developing a software is only a matter of spending a few hours a day in front of a computer. Selling 1000 copies of a software at 50 Euros through the Internet is sufficient for an independent software publisher to develop, publish and live. That's how, many independent developers live in Europe and they are quite happy from that. The success of software publishers actually depends much more on marketing issues than on technical innovation.

As a result, even in a rigid market ($\tau > \tau_1$), software publishers can sell enough copies to cover their expenses. In a matter of fact, good software publishers tend to find more features and techniques and implement them faster than big players. That's how they live and sell copies on a niche market, even if they are immediately copied.

However, if the market is too rigid ($\tau \gg \tau_1$), it is about impossible for small players to market their inventions because big players can copy features much faster than the typical latency of the market. Getting a reasonable market share before being copied turns to be impossible for small players and new entrants. **Making the market more fluid, through open standards and anti-monopolistic policies, should be considered as a priority in order to promote innovation in the software industry.**

3.4 Software with patents

Introducing patents in the software industry generates economic disorders by changing completely the business model of software publishers and reinforcing monopolistic behaviours of the market.

3.4.1 Software: complex system and low costs

The software industry has a few striking characteristics

1. Software is a complex system: small players have a very hard life to obtain fair patent licenses fees from big players

2. Very low investment is needed to make software. The total administrative costs of one patent (more than 150.000 Euros[24]) is actually more expensive that the total development costs of a full featured complex software which may actually include a hundreed patentable inventions.

3. Software is a land of trivial inventions because many patentable inventions can be found in a matter of minutes and still be very useful. Most inventions in the software industry consist in listening to customers or finding the right problems rather than finding new solutions to existing problems.

4. There are very few legitimate inventions in the software industry. Experts tend to admit that less than 10% of software patents in Europe or in the USA cover legitimate inventions.

5. The software market is very rigid due to the lack of public policy[25]to enforce competition, interoperable technical standards and high quality standards.

6. Software is a natural land for monopolies because the Law of Diminishing Returns does not apply to the technical part of software.

3.4.2 Consequences

The positive economic consequence of software patents is the following:

1. Patents help to finance a few legitimate inventions that cannot be easily self financed under the conditions of the traditional software publishing business model (3.1.6 and 3.3.2)

The negative consequences of software patents are the following:

1. Patents help to make big players stronger (3.1.7 and 3.1.9)

2. Patents tend to exclude profitable and useful small players (3.2)

3. Patents reinforce monopolies (consequence of previous paragraghs)

4. Patents increase expenditures on legal advice at the expense of research programmes (3.1.8)

5. Patents make public researchers believe that they will be financed through patents but actually force governments to finance research through tax even more (3.1.4 and 3.1.9)

On the whole, it seems that software patents are more harmful than useful. That is actually what many industry leaders in Europe believe[26]. Still, there are some positive economic aspects to software patents which require us to study how we could benefit from them without suffering from the negative consequences.

[24]http://www.lmi.fr/src/lmi/article/articleL.nsf/29218274bd4ff0ecc125677600303b39/4a50754e3732918cc12567a0004bba95

[25]Reference to Corel and Canada Association of Software publishers criticizing governments for chosing only Microsoft products. Help needed.

[26]http://www.freepatents.org/pr1.html

3.5 Other economic approaches

A few economists have an opinion on software patents.

For example, according[27] to Prof. Hal Varian of University of California, Berkeley, author of the economic best seller "Information Rules"[28], the economy of information technology (software, web services, etc.) does not need patents. Patents should be considered from a cost/benefit point of view. The economic benefit of patents is to garantee some kind of reward to entrepreneurs. The economic cost of patents is to generate administrative costs and monopolistic markets.

Because of the existence of "positive feedbacks", the information technology market tends to generate "winner take all" situations in a very short time. This means that the reward for the winner is extremely high. Introducing patents in the system has only a marginal effect because it will not increase the reward for inventors. But it will generate administrative costs and slow down standardisation.

Since patents generate more costs than benefits in the information technology market, they do not seem to be necessary, from an economic point of view concluded Prof. Varian.

4 Justice: are patents fair?

Patents may be useful or useless. But are patents fair?

4.1 Patents on legitimate inventions

Anyone will probably agree that intellectual property is fair in the following situation: someone invests a lot of money and finds an invention then goes to see an investor in order to get extra money to develop the idea. Then, the investor copies the idea and asks someone else to implement it instead of asking the inventor. This is quite unfair for the inventor since he can consider that his invention has been stolen. Patents can be understood as a fair approach to protect inventors against unfair copy. For this application, patents need not last very long: only the average time required usually to find money and industrialise an invention.

But, as we saw previously, it is necessary to give a chance to inventors to get some market share. That's why granting long monopolies to patent owners can be eventually considered as fair in order to let inventors obtain benefits from their invention.

[27] Public lecture given at BNP-Paribas, Paris, september 14, 1999

[28] Information Rules A Strategic Guide to the Network Economy Carl Shapiro and Hal R. Varian.

4.2 Patents on minor inventions

Patents on minor inventions are only profitable to big players because of market rigidity. This may sound rather unfair in terms of competition.

Big players may choose a strategy consisting in spending a lot of money to improve step by step their products by adding constantly many minor innovations and leverage a significant market share to garantee high benefits. This kind of strategy, introduced by japanese multinationals and used nowadays by Microsoft, can be very efficient to obtain and maintain a dominant position in the market, especially if small competitors are not granted patent licenses or are granted patent licenses at expensive fees.

The only way to make minor inventions fair in terms of competition would be to garantee a fair and non-exclusive access to such inventions. Still, in most cases, patents on minor inventions are not necessary for big players to cover their research expenses because of market rigidity. It would probably not be unfair, and much simpler at the same time, to decide to eliminate patents on minor inventions.

4.3 Patents on trivial inventions

It is usually accepted that ideas should not be protected by intellectual property because:

1. it easy to find tons of ideas with no risk and with little investment,
2. it is more important and more fair to give some protection to people who take risk and implement ideas rather to people who find ideas,
3. many people usually get the same ideas at the same time and it is hard to prove who is the first.

That's why only inventions can be patented. The problem with trivial inventions is that their nature is very similar to the nature of ideas although trivial inventions are not ideas, as soon as they follow the legal definition of inventions. Trivial inventions actually share most of the caracteristics of ideas listed above. And, as we saw previously, patents on trivial inventions mainly finance lawyers, patent experts and patent offices rather than for inventors.

A trivial invention could actually be defined as "an idea described by a patent experts as the technical solution of a technical problem susceptible of industrial application". This explains why it is very hard for patent offices to eliminate trivial inventions through a pure legal approach. Writing the text of a patent is nearly an art. The most skilled patent experts are sometimes proud to explain that they could get a patent on anything. The younger and less skilled patent experts usually work at the patent office. Because the most skilled people are the ones who write the patents rather than the ones who review them, the patent system will intrinsically accept more patents than it should whatever the legal definition of inventions. That's why, although it is obviously unfair,

patent offices tend to grant more and more patents on software related trivial inventions.

Eliminating trivial inventions from patentable inventions could make patents more fair. Most proposed approaches are based on economic considerations rather than legal classification. But it is not in the European legal tradition to include economic parameters in the law. And it is nearly impossible to say, without economic parameters, what a trivial invention is and what it is not. Still, if one accepts that inventivity is determined by the formulation of the solution rather than the formulation of a problem, a possible approach[29] would be to ask inventors to publish the problem and wait for one year to let people suggest solutions. If noone, within that timeframe, suggests a solution similar to the inventor's own solution, the inventivity criterion is satisfied. Otherwise, the inventor's solution is considered as non-inventive.

4.4 Patents and exclusion

Patents grant to their owner a power of exclusion.

The ownership of intellectual propety on inventions actually means that the owner is entitled to decide who can and who cannot use his invention and at what price. Granting patents to people who decide not to implement the invention or to sell it at an exaggerated high price may sound unfair for all the others who would be delighted to benefit from the invention.

As a result, the French law does not allow the owner of a patent to keep it without marketing it. After 3 years (L613-11 of French Intellectual Property Code), patent owners are forced to grant licenses to anyone who asks for a license if they did not seriously try to market it.

Some inventions are related to health as well as "ordre public". Since health and "ordre public" are considered as values than property in most civilisations, the TRIPS agreements (Article 27.2) state that national laws "may exclude from patentability inventions, the prevention within their territory of the commercial exploitation of which is necessary to protect ordre public or morality, including to protect human, animal or plant life or health or to avoid serious prejudice to the environment, provided that such exclusion is not made merely because the exploitation is prohibited by their law".

Still, patents on pharmaceutical drugs are legal in most countries. As a result, some drugs produced in rich countries by patent owners are too expensive for third world countries. On the other hand, those drugs could be very cheap to use if they were produced by third world countries without paying an expensive patent fee. Because it sounded very unfair to exclude third world countries from cheap modern medecine (ex. AIDS multiple therapies), it has been decided lately to grant third world countries the right to produce their drugs for a very low patent license fee.

[29] This approach was suggested by Hartmut Pilch (FFII).

4.5 Patents and public education

Teaching and improving the human knowledge through research are often regarded in most advanced civilisations as higher values than property. In order to prevent patent owners from prohiting teaching patented invention to students or improving a patented invention through research, national laws usually state that the rights of the patent owner do not apply to education or research[30].

4.6 Patents and brain drain

Imagine a country which trains few engineers and trains many lawyers. This country has very broad patent laws and tries to force all countries in the world to implement the same laws as well as to recognize its patents. Imagine another country which trains many engineers which get excellent fellowships paid by the taxpayers so that students are more attracted by scientific studies than by law.

Although people are not usually allowed to imigrate to the first country, an exception is made for engineers. Engineers are welcome and better paid . Many engineers from the second country decide to emigrate to the first. They work in private companies for a few years, find very nice inventions which in the end get patented and owned by the companies of the first country.

Then, companies in the second country have to pay to companies from the first country in order get licenses on patents.

If we sumarise the spendings. Citizens in the first country

1. train few engineers
2. get money from the patents found by engineers trained in the second country

Citizens from the second country:

1. pay taxes to train engineers
2. must pay to use the inventions found by the engineers that were trained with their taxes

In a situation where there is a clear brain drain from the first country on the second country, the use of patents tend to be extremely unfair for all inventions that can be found in a short time because training engineers is much more expensive than paying a few years of high salary.

4.7 Patents and complex systems

The issue that makes patents unfair for complex systems is the fact that cross-licensing policies won't usually be fair between big players and small players. It does not matter so much in industries which require a lot of investment (cars, micro-electronics, etc.) since small players can hardly survive independantly.

[30]However, this is not the case in the USA.

However, in the software activity as well as in any activity related to immaterial goods (consulting, service, etc.), there is no need for big investment in order to become a profitable and useful player. That's why fair non exclusive patent licensing policies should be enforced for complex system industries in order to make patents more fair.

4.8 Patents and monopolies

Patents make monopolies stronger because

1. it gives big players more power of exclusion against the rest of the players,
2. the bigger the player, the more money it can pay for an exclusive license and the more profit it can get from an exclusive license.

Fair and non exclusive licensing policies would be required in order to make patents more fair in a situation with monopolies so that any company could get a license at a fair price, not only the monopoly. As a result of anti-trust laws, this kind of policy has been implemented in the USA in the case of IBM: anyone may get a non-exclusive license on the hole patent portfolio of IBM for 5% of its revenue[31].

5 Software Useright: useful and fair patents for the software industry

Most innovation in the software industry have been either self financed (case of applied research made by most software publishers in Europe) or financed by public funds (case of fundemantal research held at public facilities) without the use of software patents.

The main justification for introducing lately patents in the software industry is to allow the private funding of research projects which cannot be self financed. Other acceptable justifications include:

1. Encourage people who develop web sites to publish their inventions
2. Encourage people who develop software for internal use to publish their inventions
3. Encourage software publishers to use patents instead of source code secret to protect their investment

However, patents in the software industry raise many concerns:

1. They could actually encourage some players to stop publishing their source code

[31] Initially, the amount was 3% but it has been increased recently

2. They put at high risk software assemblers, software distributors and software service companies
3. They are very unfair for small players
4. They could restrict interoperability
5. They reinforce emerging monopolies
6. They generate administrative costs which will lead to higher consumer price

It is therefore necessary to look for solutions in order to make patents more fair and useful

5.1 Patents on programmes as such

First, programmes as such should not be considered as patentable inventions because:

1. it would generate many juridical contradictions, including with copyright
2. it would allow to patent business methods and social practices through patents on programmes which implement innovative business methods and social practices
3. it would put software publishers, software and distributors at very high legal risk whenever they include in their product some patented software technique without even being aware of it

We recommend instead stating explicitely that inventions involving computer programmes are patentable and at the same time to explain the principle of "copyright vs. useright" described bellow.

5.2 Software related inventions: new information processes susceptible of industrial application

Programmes as such should actually be considered as a description of a software related invention which can be understood by a computing device, just as a patent document is a description of an invention which can be understood by a human being.

Software related inventions should be of course new, inventive and susceptible of industrial application. In particular, software related inventive techniques not susceptible of industrial application but susceptible of social application are not inventions and, thus, not patentable.

This leaves as a possible defition for software related inventions the following: "new information process susceptible of industrial application" where the

information process should be described as the technical solution of a technical problem. With this definition, computer programmes as such cannot be patented because programmes are not processes but descriptions of processes.

However, a computer programme can become a part of a patented process once it has been read, understood and executed by a computing device. This is similar to a human being reading a patent document and implementing it.

Although the nature of a process is immaterial, it cannot be considered as work. Copyright law does not apply to information processes. Only patent law can.

5.3 Copyright vs. Useright: the right to copy vs. the right to use

The definition of software related inventions as information processes allows to remove most contradictions between copyright law and patent law. It makes a clear disctinction between the right to copy a software and the right to use a software.

1. Copyright law is used to define the right to **copy**, modify and adapt a software which includes the right to copy, modify and adapt a computer programme. Copyright infringement happens whenever a specific software is copied without authorisation from the copyright owner.

2. Patent law defines the right to **use** a patented information process. Patent infringement does not happen when copying a software. However, patent infringement happens if one uses a software in such way that a patented process is executed by a computing device without the authorisation from the patent owner.

This clear separation allows to make the situation less risky for software publishers, independent software developers, service companies, software distributors, free software, etc.

Still, it does not mean that patents cannot enforced. Software publishers who are aware that they are providing to users something which enables the user to use a patented information process can be made responsible for that in some way for contributory infringement (*fourniture de moyens* in French).

5.4 Patent keys

Users may require a way to be certain that they are not infringing on a patented process by using a programme. Software developers may want to be certain that they are not contributing to an infringement. And inventors of information processes may want to get some reward quickly and easily.

In order to achieve the three goals at the same time, we suggest a key system for software patents, which would apply to both proprietary and free software. Keys are implemented as individual encrypted files. Each key represents the right for a user to use a software related patented invention.

Software developers are encouraged to include a "patent key" function which only allows to execute the software, thus potentially creating instances of patented processes, if the necessary keys have been acquired by the user or provided together with the software by the publisher, the distributor or the service company. The "patent key" function protects software developers from contributory infringement claims.

The "patent key" function may eventually check a central database of patented inventions in order to make sure that all patent claims for a software are known. This protects users from being sued for unknown infringements.

Inventors of software related inventions should be encouraged to register their inventions on a "patent key" server for license fees collection . Private companies may also offer various kinds of contributory infringement searching services to help inventors and increase their license revenues.

Keys for non commercial and private use should always be free (since patents do not apply to non commercial private situations). The same rule stands for research activities. On the other hand, keys for commercial use should be paid to a central patent fee collection authority which redistributes license fees to users.

To make things easier for users, software publishers and software distributors may decide to sell different versions of a software, some including patent keys for commercial use, others including patent keys for non commercial use and others including no patent keys for users who have already acquired the keys.

In case of patent conflict due to the discovery of a patented invention in a software after the software has been published, the software publisher and the "patent key" authority may either decide that:

1. each commercial user should be asked to pay in order to keep on using the software;
2. or that the software publisher buys licenses for each of its users.

Software users would then know who is asking for money, and for what kind of invention (legitimate, minor or trivial). This would put a strong pressure on people who file patents on minor or trivial inventions and then ask everyone to pay for little used features or worthless administrative costs.

Incidentally, patent keys solve the requirements of article 100A of in terms of consumer protection.

5.4.1 Patent key market

Patent keys should be traded on an electronic market between patent producers, patent users and patent publishers. This would make it difficult to sell licenses on patents at very different price to one software publisher and to another.

5.5 Interoperability

Licenses on software related inventions should be free or fair, and granted without delay, for interoperability purpose.

5.6 Restricting patents to legitimate inventions

It is necessary to restrict the number of patents on trivial inventions. Moreover, patents on minor inventions are not necessary. Restricting patents to legitimate inventions is probably the way to go.

One way to do this is to ask patents to have a clear industrial application. Another approach would be to include some discriminiation based on economic evalutation of inventions as it has been proposed by Germany a few years ago[32]. But, although it is not illegal, it is in contradiction with the European legal tradition.

Another approach (4.3), requires to accept that inventivity is determined by the formulation of the solution rather than the formulation of a problem.A possible approach[33] would be to ask inventors to publish the problem and wait for one year to let people suggest solutions. If noone, within that timeframe, suggests a solution similar to the inventor's own solution, the inventivity criterion is satisfied. Otherwise, the inventor's solution is considered as non-inventive.

5.7 Open sourcing

A situation where some software are published with source code, where other software are published without and where patents can be enforced is a strong incentive not to publish source code any longer. This would be very damageable for health and security (2.9). The following approaches should be considered in order to make the EU patent law fully compliant with the requirements of article 100A of the Rome treaty in terms of high level of security:

1. update the software copyright law and include some clause stating that all source codes should be made available at no cost for patent contributory infringement searching;

2. restrict the rights granted by software patents so that open source software can never be considered as a contributory infringement.

The first approach generates both an increase in terms of safety and an increase in terms of consumer protection wich is fully consistent with article 100A. The second approach does not change the current level of safety or consumer protection.

5.8 Fair licensing

The most questionable part of software patents for their economic utility is that fair licensing policies cannot be obtained between big players and small players. Two approaches can be considered:

[32] Ask DiCosmo reference. Help needed.

[33] This approach was suggested by Hartmut Pilch (FFII).

5.8.1 Regulation

Licensing of sub parts of a complex system should be automatic after 2-3 years and at a fair price decided either by the parties or by an independent pricing authority. This could be an extension of the law stating that "non exclusive licenses should be granted automatically if the patent was not implemented and marketed" by considering that inventions should be available individually as generic independent components in order to be considered as available to the market rather than bundled. For example, if Rolls Royce refused to grant to other car makers some patent licenses on specific inventions, one may wonder if the invention is really available to the general market.

5.8.2 Battle

If regulation laws can't be obtained, it is possible to get the same effect through mutual patent funds managed by a worldwide coalition of independent software developers. The principle of patent funds are based on mutual property on patents with various subtle clauses to protect the coalition.

5.9 Promote SMEs

Last question: what can be done for brain drain?

Some governments in Europe are setting up rules forcing students to keep secrets and to give their intellectual property to universities[34]. The actual practical consequences of such rules in the software industry is that students will be even more encouraged to leave Europe to the USA since very few universities are going to sue a company in the USA.

Moreover, software patent funds in the public sector usually generate more losts than profits because public institutions are not always able to market and protect properly their property. The only situation where public institutions may use their patents would be to sue a small company created by a former student in his country, close to the university, because it's easy, cheap, and has no consequences on the contracts the public institution may have with big national companies. Again, that's one more reason to leave Europe.

If governments really want to protect the taxpayer's investment in public sofftware research and development, they should consider of the following approaches:

1. Make education fees free for students if and only if they accept a contract stating that they are going to work for their country for at least 10 years (many students in France have to sign such contracts in order to get a fellowship).

[34] A PhD student at Paris VI university who was financing his PhD himself with no fellowship has been asked recently to sign an NDA paper and accept to transfer all intellectual property of his work the the university without counterpart.

2. Encourage students to create their own company in their country. In the case of software, the best way to encourage someone is to provide him with a contract rather than with subsidies.

Conclusion

Software patents are generally useless, worthless and unfair. Patents on programmes as such are even dangerous because they allow to grant monopolies on business methods and social practices and make business life very risky for software publishers.

However, two situations may require software patents: private funding of fundamental research and the increase of web applications which are kept secret. Still, patents are only efficient in competitive markets with no monopolies which is not the case of the software industry because of positive feedback effects.

The clarifications of the software patent law in Europe should therefore give us an opportunity to introduce various regulations in order to stimulate interoperability, competition and SMEs. Such regulations include non exclusive and fair licensing policies, a clear distinction between the right to use (useright) and the right to copy (copyright) a software, electronic keys cof software patents, obligations to access the source code of software, etc.

If such regulations are included, then software patents, as well as utility certificates for software, may become more useful than harmful. Otherwise, software patents are going to be more harmful than useful and may even force governements to spend much more taxpayers' money to support public research in computer science than in a situation without software patents.

Annex - Towards a computational model

In order to study more in detail the economic consequences of patents in the software industry, we suggest for a future paper to implement a behavioural simulation of the economy of software patents.

Implementation

The idea is to use a high performance parallel computer such as a Linux Beowulf cluster and throw into it at a constant rate thousands of autonomous actors[35] with different predefined behaviours. The environment is defined as a set of rules which apply to all actors. Some actors tend to grow, other are going die.

After a certain time, unexpected communities of actors and common behaviours may eventually appear in the system.

[35] An actor in computer science is an active object.

The noosphere

The noosphere contains an infinite number of inventions. Some are useful, some are useless. Some can be used immediately, some not due to the evolution of raw computer power (ex. ray tracing, speach recognition can only be used in the mass market today, not 10 years ago).

Each economic actor needs to pay a certain amount of money to extract a new invention from the noosphere. Each extracted invention has a potential benefit b for potential consumers in y years ahead. There is no direct relation between the cost of inventions and the benefits they provide.

Once they get an invention, actors may

1. Publish the invention
2. Keep the invention secret
3. Implement the invention
4. Not implement the invention
5. Patent the invention
6. Not patent the invention

Patenting the invention requires expensive administrative costs.

Publishers and patents

We then consider a group of companies with the following behaviour

1. Some companies implement the invention if it is useful and follow a software publisher model. This is of course somehow irrational in world where patents can be sold to big players at a higher price than the profit one could make by developing software. Still, many people are happy to develop software.
2. Some companies patent inventions and try to sell the patent
3. Some companies buy exclusive licenses from patent traders

Anyone can use the inventions of type 1. Companies of type 3 tend to cross-license their inventions on a one to one basis. Companies of type 1 tend to try to bypass inventions of type 3. In order to so, they spend at least as much as the development costs of companies of type 3.

Expected results

It would be interesting to see the results of such a simulation:

1. Who is going to win?
2. Will free processes replace proprietary inventions?

The results of this simulation may be covered in a future paper.