SOFTWARE PATENTS AND INNOVATION

Sylvain Perchaud

Abstract: The European Commission has put software patents on its agenda. For the first time in Europe's history a monopole will be granted by the State on immaterial ideas. The goal of this paper is to introduce the reader to the particular economics of the software sector and to compare the costs and benefits of a market with software patents against the actual situation. For this purpose we'll explain which were the crucial elements that enabled the birth of the Silicon Valley, what is the actual set-up of the european software industry and if patents are needed by the players of this market.

Introduction

The European Commission is writting since the summer of the year 2000 a directive on the harmonization of the patent offices practices according to the Munich Convention on the European Patent of 1973. At first see reserved for the lawyers and counsellors in intellectual property, the debate is also intense among the economists since the European Patent Office (EPO) clearly shows a will to take off the computer programs from the list of exceptions to patentability, defined in the Article 52 of the Convention. The advanced arguments for the patent protection for softwares, although they are already protected by the copyright system, are mainly of a juridical level (the EPO proclaming that the patentable inventions are the technical solutions to technical problems) and political level (according to a certain reading of the TRIPS Agreement, Europe is compelled to take off the softwares from the lists of non-patentable inventions).

Now the european and american economists have reserves on the cost of such software patents for the competition and even the innovation in the software sector. In fact the introduction of software patents in the USA doesn't seem to have produced the awaited effects (incentive to innovate), and the flaws of the american system has led to abuses (the competition between software publishers is moving from the market place to the Courts) threatening the freedom to innovate and competition. It is particularly interesting to notice that we've applied in the USA the patent system as such, without taking into account at any moment the specificities of the software market as the inner nature of innovation, the research and development costs, the duration of innovation cycles, the need for standards and strong network effects...

Thanks to the latest progress of the economic analysis in terms of dynamic models, measure of transaction costs or even of industrial strategy, the role of the economist is becoming crucial in order to calculate the costs and benefits of the steady growing we actually observe in the industrial and intellectual property sphere. A property system as morally justifiable as can be can' be in any case accepted if its operating costs are superior to its benefits for the society taken as a whole.

So, what is so specific to software so that the European countries (but also other nations with a strong IT sector such as Canada) decided thirty years ago to make software non-patentable ?

2. Innovation in the software sector

2.1 Process of creation

To understand what a software is, it's important to study how do we create one and what are the means we've to use to produce a final version of the product.

A software is a set of logical instructions interpreted by the computer, which one produce a digital result.

In order to avoid to the coders to use binary code (which is the inner language of the

computer, 0 and 1 digits being the only symbols available), programmation languages were created to help the creation of softwares with a clear and easy to read logical “grammar”.

Therefore the work of the coder is to write the right instructions, variables and algorithms in his/her software so that the computer reacts as planed.

We can compare the work of the coder to the work of the architect. As in the case of the architect, the quality of the result depends less on the bricks (here, the algorithms) than on the logic of the whole work (the software), even if the bricks are the constituent elements of the creation.

The ties between programming and mathematics are then very strong.

2.2 Patents and sequential innovation

The software has historically had a week intellectual property protection and has known an extremely quick imitation. Nevertheless the software sector is one of the most innovative of the whole economy.

To understand this undeniable fact, we have to study how innovation appears for this kind of product ; we’ll use for this purpose some historical examples.

A modern software is today composed of at least a few thousands or hundred thousands of ligns of code (source code). All these ligns of code are use by the program to manage the I/O (for example to show on screan what is typed on the keyboard), to react to events (which function to execute when the user click on an element...), to use an algorithm to process the information (retouch effect on a picture, calculation in a spreadsheet...) and in certain cases to interpret languages or scripts (HTML for browsers, task automation with AppleScript, shell scripts etc...).

A software is a unique arrangement of bricks, which are algorithms. Sharing and re-using bricks is therefore common and allows to achieve an optimum in the management of time during the creation of a new software.

This is with this re-use of already existing bricks that we find the sequential nature of innovation in the software sector[1] .

Thanks to the Fraunhofer Institut’s study[2] conducted last year among the german players, we see with these data that the reuse of code is an important element during the creation of a new software. Even one third of the softwares contains more than fifty percent of existing code.

The history of the main softwares perfectly shows this inner property of innovation in the software sector ; each new software has introduced new innovations on top of the

[1] J. Bessen and E. Maskin (2000) “Sequential Innovation, Patents, and Imitation”, MIT

[2] Max Planck Institut / Fraunhofer Institut (Septembre 2001), "Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen : Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation". <http://www.bmwi.de/Homepage/download/technologie/Softwarepatentstudie.pdf>

previous innovations, no software was created *ex nihilo*.

A well known history is the one of the web browsers. When in 1989 the web was invented in the CERN (european center of nuclear research, based on the french-swiss border), it was decided that the pages will be formated according to the HTML language. To interprete and display these pages a text browser was created: Lynx. While being rudimentary, it was conceived to be used by scientists on Unix workstations and fullfiled its goal. Then the HTML was enhanced to include graphical data and Mosaic came up with an intuitive interface and the ability to display graphic data. Then Netscape (a Mosaic with a few new features) appeared and quickly became the leading browser. Seeing the browser as a threat to its monopole, Microsoft reacted and launched its own browser, Internet Explorer. To develop quickly its browser, the software giant simply bought the code of Mosaic and hired lots of Netscape employees. Internet Explorer was nothing more than a Mosaic-plus.

The lesson of this history (could have been the history of spreadsheets with Lotus 1-2-3 and Excell or any other kind of software) is that the innovation is made sequentially, each new software comes with its own new features on top of the existing others[3].

Moreover the rate of innovation (in less than ten years we passed from the text based browser such as Lynx to today's user friendly, multimedia browser such as Opera) doesn't correspond at all to the patent monopole duration (which lasts twenty years).

With such an incompatibility between the cycle of innovation and the patent protection we can face a *tragedy of the anticommons*[4] where we have too many patent owners. In this situation too many private individuals hold rights on blocks of innovation that constitute obstacles to future research, we see a large increase of costs of transactions since each fragment of a software is held by a different owner ; only large companies with deep patent portfoliols will be able to avoid the tragedy of the anticommons by creating patent pools between them (this can be seen as a breach of european antitrust laws). Less innovation will occur since the small and innovative players won't be able to get the rights for each block included in their software for the following reasons:

- there's no compulsory licensing, therefore no incentive for the patent owner to grant a license to an ingenious competitor.
- the cost in time and fees is too high compared to the average software life cycle (3 years).

3. Patents and cycle of innovation

As we've seen in the previous example, the cycle of innovation seems to be far shorter in the software sector than in industries producing physical goods. This assertion is corroborated by the figures of the Fraunhofer Institute study[5], in the german market more than seventy percent of the companies develop their softwares in less than twelve months.

[3] for a further econometric study, see J. Bessen and E. Maskin (2000) "Sequential Innovation, Patents, and Imitation", MIT

[4] M.A. Heller, R.S. Eisenberg (1998), "Can patents deter innovation ? The anticommons in biomedical research", Science magazine vol. 280.

[5] Fraunhofer Institut's study, see graph 4.2.4 "Durchschnittliche Entwicklungdauer von Softwareprodukten" page 62

Even more important: more than forty percent only need six or less months to develop a software, showing a clear difference between this sector and the physical innovations (mecanics, medics...).

The life cycle of the product in itself is also very short, more than seventy five percent[6] of the customers are replacing their softwares each year and almost fifty percent are replacing them every six months. This shows a true dynamic, encouraged by the software publishers (the economists have studied this behaviour as the planed obsolescence).

Needless to say, software innovations are made available on the market before their creator is granted a patent protection. It takes an average delay of thirty four months to get a patent in the USA and eighteen to twenty four months to get a patent in Europe, that is to say that if you applied for a patent, the innovation you wanted to protect is already outdated by your competitors and the customers (on a free market) have decided to embrace or to reject your software. The market has decided the fate of your product[7] .

Moreover, as we've seen in 2.2, a software is made of hundred or even thousands of algorithms. Companies such as IBM are filling more than five hundred software patents per year in the USA[8] (in 1995 more than seven thousands software patents were issued in the USA), and the matter of many of these patents is obvious and doesn't make any technical contribution to the state of the art[9].

So any new software is built upon previous innovations and is likely to infringe on many patents, since many basic software elements or algorithms are already patented (Palette patent from Adobe[10] , LZW compression algorithm patent[11] which covers *.zip*, *.gif* and *.pdf* files, etc[12]). Since software companies don't have the time nor the necessary knowledge to browse patent databases to check if each algorithm inside their computer program infringe on a patent claim, they will have to hire patent experts.

This cost isn't only a financial cost, but also a time-to-market cost since the R&D department (the coders), for each new line of code, will have to ask to the patent department to look for prior art in the patent databases. Once the patent experts have found which algorithms are already patented will have to discuss with the patent department for each new line of code.

Another reason why software patents are dangerous for the independent developpers is that the way the claims are written are non-understandable for the average coder. Not only the patent doesn't give at all the only useful information that is the source code but it can be written in such a broad way that other technical solutions are also infringing the claim. Therefore patenting is more and more about finding way to descript a problem than

---

[6] Fraunhofer Institut's study, page 63
[7] see software innovation cycle graphic in the appendix
[8] J. Bessen, E. Makin "Sequential innovation, Patents, and imitation", page 24
[9] G. Aharonian, P. Hartmut...
[10] Patent number EP0689133 ; see also <http://swpat.ffii.org/vreji/pikta/xrani/palette/index.en.html>
[11] <http://swpat.ffii.org/vreji/pikta/xrani/gif-lzw/index.en.html>
[12] See the FFII database: <http://swpat.ffii.org/vreji/pikta/xrani/index.en.html>

to give the solution to the problem, making impossible to avoid the patent with a different solution. The only way to solve this problem is to compel the patent owner to publish the source code in its patent claim.

4. Creativity and reactivity of micro-structures

During the sixties and the seventies the softwares were mainly created by the hardware manufacturer, the scientists or the academic researchers. The offer of commercial softwares was non-existent and the companies, banks, insurances and other main sectors were developping themselves their own computer applications.

Then during the end of the seventies the first personnal computers appeared. With this democratisation of computers the first independent software publishers were founded, among them the leading players of today's sector: Microsoft, Lotus or Adobe.

As we've seen in the previous chapters the creation process of a software is only logic, there's no physical processes to use. So the necessary capital to publish a software is minimal, you only need a computer and enough money to pay for the electricity and telecom bills.

It is therefore not surprising that the world center of software innovation is the Silicon Valley were the economy is one of the most liberal and were many young graduated enginers are living. The easiness of the founding of a company has allowed young students owning a few dollars to publish their creations by taking minimal financial risks.

Unlike physical industry where variable costs are a major part of the total cost of the product, the production costs are near zero in the case of software, and this is even more true today than before (thanks to the internet the transportation and duplication costs are taken in charge by the customer). There's no logistic (management of stocks of products, transport costs, duplication costs...).

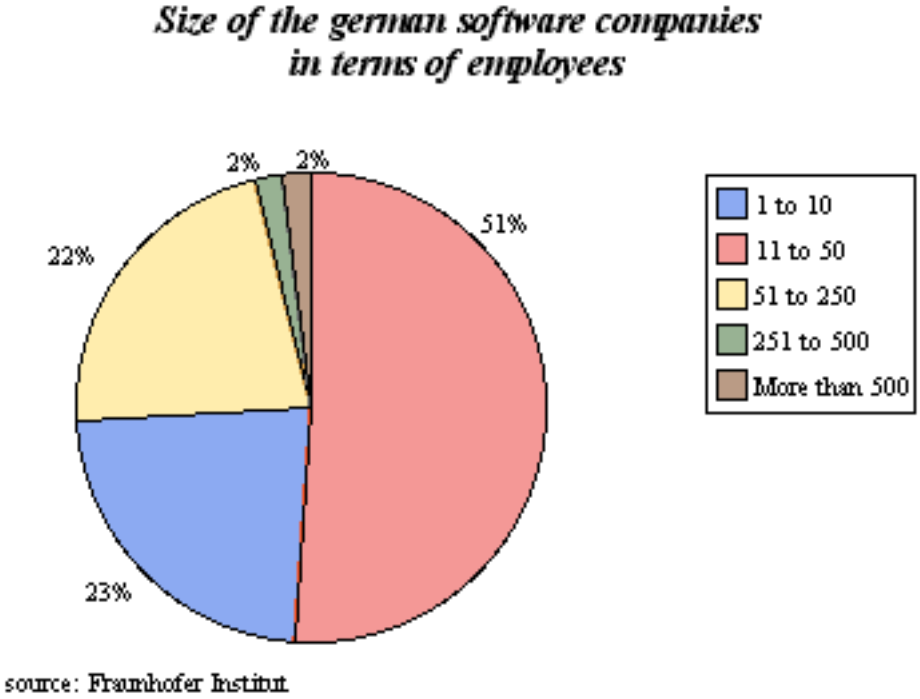
The only remaining cost is the fixed cost, that is to say the cost of designing, developping and debugging the software.

The constituents of such a cost are mainly time and the training of the employees, the cost of buying a computer being insignificant (less than 1200 € per machine). To make it short, in order to publish a software you only need the necessary knowledge in computer science, a computer and time at hand ; this picture exactly correspond to young students.

It is therefore not surprising that in a 20 years old sector, made of giants such as Microsoft, SAP or Oracle, the main innovations still come from micro-companies founded by students. There are tons of examples :

- the democratisation of the internet with Netscape
- data compression with the shareware Winzip
- the MP3 democratisation and skins interfaces with the shareware Winamp
- P2P with Napster
- the DivX video format, made by a young french
- etc...

This assumption is confirmed by the statistical surveys conducted in Europe. As we can see in the following graphic:



As indicates this graphic created with the results of the Fraunhofer Institut[13] study, almost 75% of the german software publishers count less than 50 employees, and only 4% employ more than 250 people. The same results appear in France[14] where 58% of the companies count 2 employees or less and only 1,3% count more than 100 employees. The Fraunhofer Institut's figures can be extended to the whole european software sector, Germany being the leading producer of softwares and the first market in Europe.

As a conclusion for this chapter we can say that the micro-structure is the most frequent kind of firm in the software sector, the entrepreneurs needing mainly human capital but don't have to care about factories or logistic. It is therefore not surprising to see a 19 years old student launch the P2P revolution with Napster.

With such an industrial set-up software patents will create a non-natural barrier for the entry of new competitors in the software market. Micro-structures don't have the necessary patent department nor the liquidity to file patents instead of large companies such as IBM which have already thousands of patents granted in the USA and will only have to file the same patents in a fast procedure to the European Patent Office.

The situation will be that the medium sized innovative european companies will be compelled to innovate and wait to see their patents granted while the american players won't innovate but simply use their existing US patent portfolio in Europe.

[13] Max Planck Institut / Fraunhofer Institut (Septembre 2001), "Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen : Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation". <http://www.bmwi.de/Homepage/download/technologie/Softwarepatentstudie.pdf>

[14] Commissariat Général du Plan (2002), "Économie du logiciel : renforcer la dynamique française".

As the average cost of patent litigation is about half a million euros[15] the european innovators won't even have the opportunity to find prior art to defend themselves, they'll in general stop the development of their software and leave the market[16].

The patent system is actually designed for large companies because of long delays to get a patent granted, high fees, long and costly trials to enforce a patent...

Moreover the value of a single patent is decreasing because many patents are now granted on obvious or already existing innovation[17] , so companies are compelled to file many patents on the same technology and only multinationals with deep pockets can do that.

Considering these facts, the software patents are therefore incompatible with the european software market set-up.

5. Patents and loss of consumer well-being

As a patent is a monopoly granted by the State, its owner can therefore sell its invention at higher prices to amortize the research and development costs.

As there's no solution of compulsory licencing in the case of software patents, the patent owner has no interest in selling a license to a competitor that is more price-competitive or innovative than him, this reduce the *ecology* of the market which is restricted to one player.

That way the price of the innovation isn't decided by the market but by the patent owner. In such a case the price is a monopoly price (the price is higher than a market price so that the company takes all the well being of the consumers) and the company takes the choice to under-supply the market.

So patents have a negative impact on the consumer well-being since, due to under-production and higher prices, many customers won't have the opportunity the use innovations than in a free market system they would have bought and taken advantage.

Conclusion

The european software market is young and mainly composed of very small companies (less than 50 employees) but is already having a great impact of the gross national product of european countries. Knowing the power of the network effects in such a market (interoperability issues, need for standards...) we need to protect and help the european startup against the mature and large american companies, some of them having even been

[15] USPTO statistics (2000)

[16] Patent on JPEG suddenly appears and forces companies that have been using it for ages to discontinue it (like Sun's Java) or pay tribute (like Sony).
News article at The Register <http://www.theregister.co.uk/content/4/26272.html>
Equivalent European Patent from the EPO public database
<http://l2.espacenet.com/espacenet/bnsviewer?CY=ep&LG=en&DB=EPD&PN=EP0266049&ID=EP+++0266049B1+I+>
Software patents horror gallery : <http://swpat.ffii.org/vreji/pikta/index.en.html>

[17] étude Greg Aharonian, article du Monde

condemned for unfair practices (antitrust procedure against IBM in the seventies and actual antitrust procedure against Microsoft both in Europe and the USA). As the Silicon Valley model has shown, venture capital is necessary to have a flourishing software industry and Europe has to develop even more its venture capital market, that is the main challenge for the coming years. Patents are monopoly rights granted by the State that lessen competition on the market, induce higher prices and slowdown of innovation, and encourage cartel behaviours (patent pools etc).

As long as the patent protection is so different compared to the innovation cycle (twenty years against three to five years) the wisdom tells us to refuse software patents. The simple fact that the result of the European Commission's survey about the need of software patents was so clear (more than 90% were against) and that the american and european experts[18] also oppose such a protection for softwares should encourage Europe to create a *sui generis* protection, with a far shorter duration, suited for softwares and business methods.

References

1. Economic documents

Aharonian G., "Patent System is Intellectually Corrupt".
<http://www.bustpatents.com/corrupt.htm>

Bessen J. et Maskin E. (2000), "Sequential Innovation, Patents, and Imitiation", MIT.
<http://www.researchoninnovation.org/patent.pdf>

Commissariat Général du Plan (2002), "Économie du logiciel : renforcer la dynamique française".
<http://www.plan.gouv.fr/organisation/seeat/Economiedulogiciel/Documents/rapport.pdf>

Dupuis Y. et Tardieu O. (2001), "La brevetabilité des logiciels", École des Mines de Paris.
<http://www.ecole.org/2/RT241001_memo.pdf>

D. Friedman (1999), "Clouds and Barbed Wire: The Economics of Intellectual Property".

Max Planck Institut / Fraunhofer Institut (Septembre 2001), "Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen : Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation".

[18] <http://swpat.ffii.org/archive/quotes/index.en.html>

<http://www.bmwi.de/Homepage/download/technologie/Softwarepatentstudie.pdf>

Jean-Claude Tarondeau (1998), “Le management des savoirs”, PUF.

Richard F. (1998), “Recherche, Invention et Innovation”, Economica.

C. Shapiro, H. Varian (1998) “Information Rules: A Strategic Guide to the Network Economy”

Smets J.P. (1999), “Software Useright: Solving Inconsistencies of Software Patents”. <http://swpat.ffii.org/vreji/prina/useright.pdf>

Smets J.P. (2000), “Stimuler la concurrence et l'innovation dans la société de l'information”.

Somaya D. et Teece D.J. (2000), “Combining Inventions in Multi-invention Products: Organizational Choices, Patents, and Public Policy”, Berkeley. <http://www.rhsmith.umd.edu/lbpp/dsomaya/Papers/DSDT00.pdf>

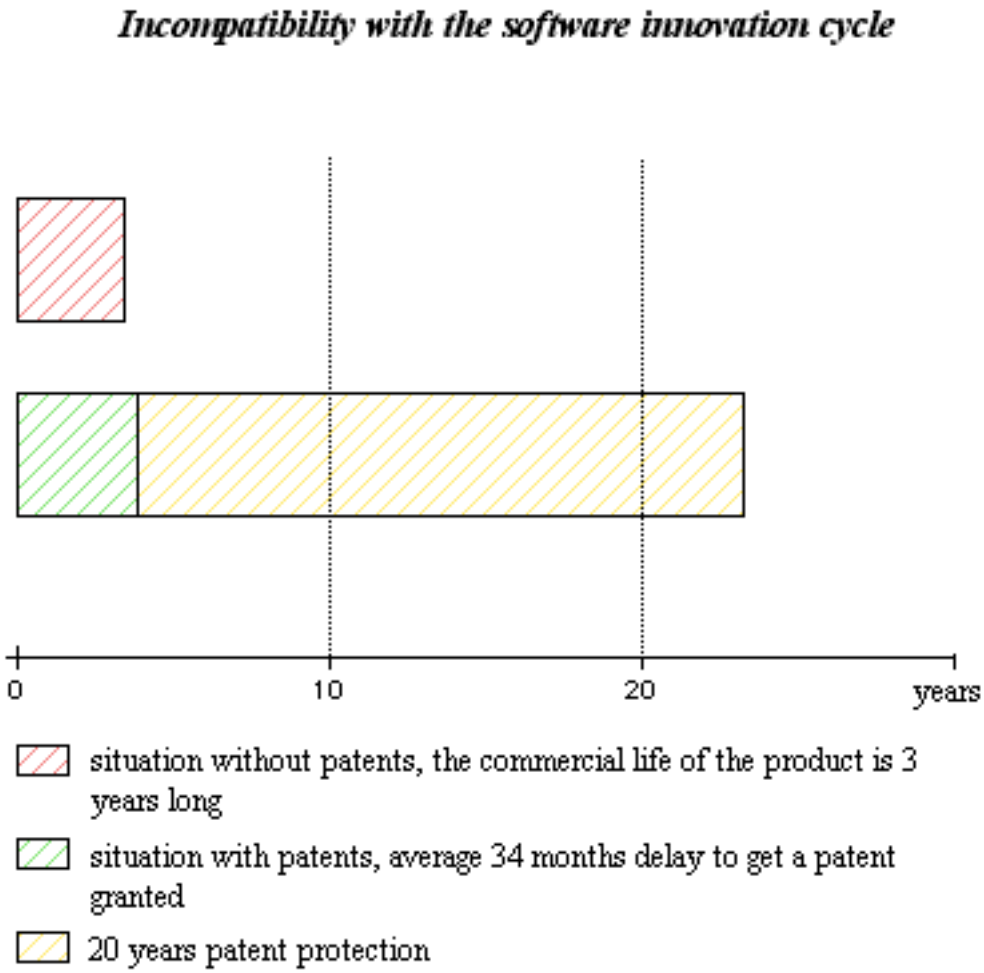
2. Law documents

DIRECTIVE COM(2002) 92 DU PARLEMENT EUROPÉEN ET DU CONSEIL concernant la brevetabilité des inventions mises en œuvre par ordinateur (2002).

FFII & Europe Shareware (2002), “Proposition(s) BSA/CCE et Contre-Proposition Basée sur la CBE”. <http://swpat.ffii.org/papiers/eubsa-swpat0202/prop/index.fr.html>

Appendix:

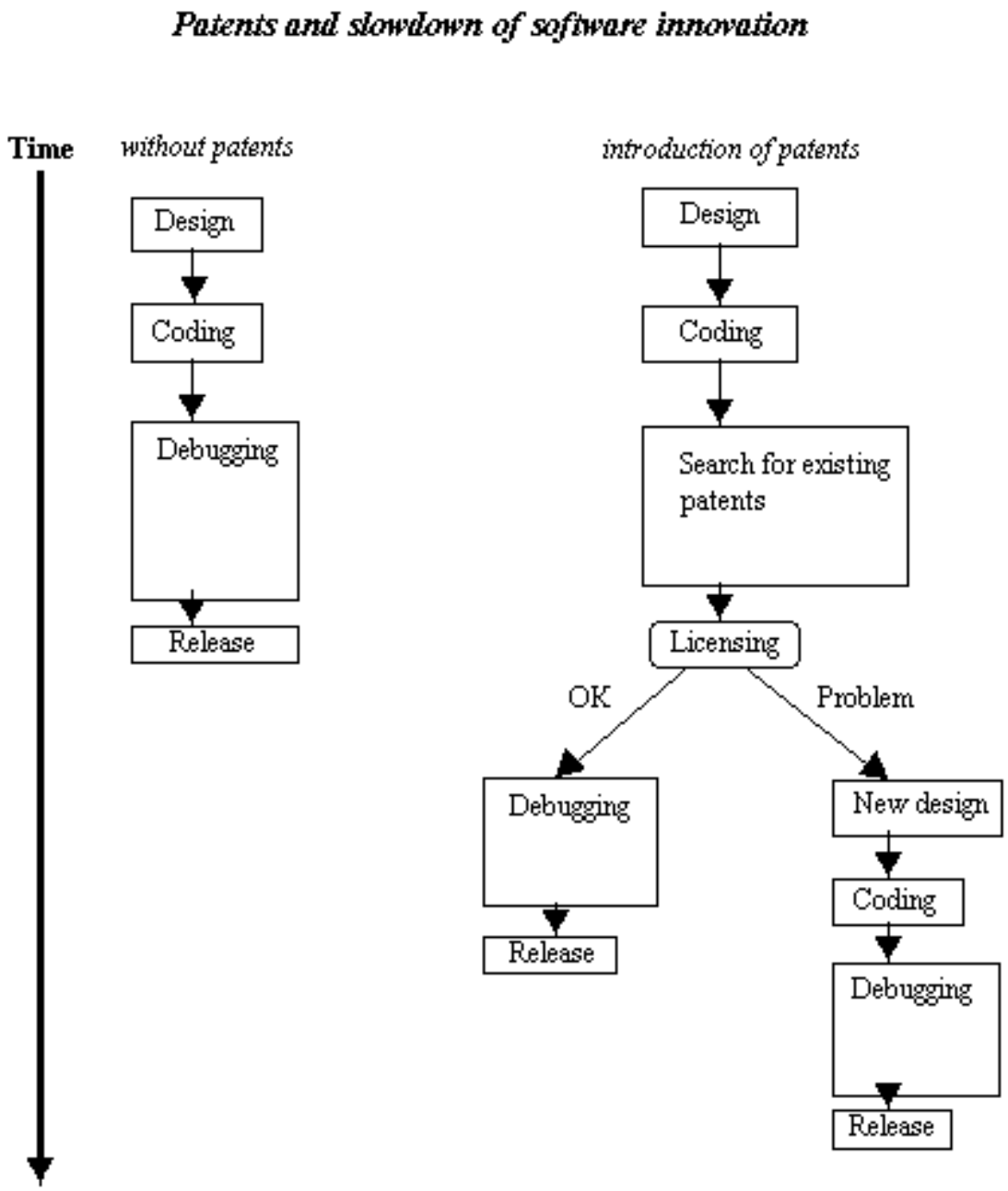
Graphic 1: software innovation cycle and patents



The delay to get a patent granted is longer than the commercial life of the innovation. This means that by the time you receive your patent the market has already decided the fate of your innovation. There's no protection for the small company.

As the patent protection doesn't match the life cycle of softwares this means that almost 6 generations of innovations can be blocked by the patent owner. This effect is contrary to the philosophy of giving incentives to innovators.

Graphic 2 : Impact of software patents on the development process



We see on this synthetic graphic the differences between the actual way of publishing software without software patents and the case with software patents.

To be as simple as possible we haven't included the impossibility to circumvent the patent(s) with the new design (that would mean that the project has to be dropped) and we've summarized the licensing discussions to a very short period of time.