Journal of Information, Law and Technology

Software Patents and Innovation

Sylvain Perchaud
President of Europe Shareware:
Association Pour la Promotion des Auteurs Européens de Sharewares

sylvain@europe-shareware.org.

This is a **refereed** article published on: 4 July 2003.

Citation: Perchaud, S,'Software Patents and Innovation', 2003 (1) *The Journal of Information, Law and Technology (JILT)*. <http://elj.warwick.ac.uk/jilt/03-1/perchaud.html>

Abstract
The European Commission has put software patents on its agenda. 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 current situation. For this purpose, we propose a perspective on the crucial elements that enabled the birth of the Silicon Valley and on the actual set-up of the European software industry. It is argued that patents are not needed by innovative players in this market.

1. Introduction
Since the summer of the year 2000, the European Commission is writing a directive on the harmonization of patent offices practices under the European Patent Convention (Munich, 1973). Initially the debate seemed reserved to lawyers and counselors in intellectual property; but soon economic consideration entered the fray as the European Patent Office (EPO) showed a will to take computer programs off the list of exceptions to patentability, defined in Article 52 of the Convention. The arguments advanced for the patent protection of software, although already protected by the copyright system, are mainly of a juridical kind (the EPO proclaiming that patentable inventions are technical solutions to technical problems) with political dimensions (according to a certain reading of the 1994 TRIPS Agreement of the WTO, Europe is compelled to take software off the lists of non-patentable inventions).

European and American economists have expressed reservations about the effects of software patents on competition and even on innovation in the software sector. In fact the introduction of software patents in the USA does not seem to have produced the awaited effects (incentive to innovate), and the flaws of the American system has led to abuses (competition between software publishers is moving from the market place to the Courts) threatening the freedom to innovate. It is particularly interesting to notice that the USA have applied the patent system as such, without taking into account the specificities of the software market as to the inner nature of innovation, research and development costs, the duration of innovation cycles, the need for standards and strong network effects...

Thanks to recent developments in economic analysis, for example in dynamic modelling, transaction costs and industrial strategy, economic considerations are becoming crucial in order to calculate the costs and benefits of the steady expansion of the sphere of industrial and intellectual property. A property system cannot be accepted as morally justifiable if its operating costs are higher than its benefits for society taken as a whole.

So, what is so specific about software that 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 software is, it is important to study the process of creation and what a developed versions are used as products. Software is a set of logical instructions interpreted by the computer which produces a digital result. In order to enable programmers to avoid using binary code (which is the inner language of the computer, 0 and 1 digits being the only symbols available), programming languages were created to help the creation of software with a clear and easy to read logical 'grammar'.

Programmers write instructions, variables and algorithms generating planned effects. We can compare the work of the programmer to the work of an 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. Thus the ties between programming and mathematics are very strong.

2.2 Patents and Sequential Innovation

Historically, software has been subject only to weak intellectual property protection and has known extremely quick imitation in the market. Nevertheless the software sector is one of the most innovative of the whole economy. We can study software innovation from historical examples.

A modern software program is composed of thousands, if not hundreds of thousands lines of code (source code). All these lines of code are used by the program to manage the I/O (for example to show on screen 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.).

Software is a unique arrangement of bricks, which are algorithms. Sharing and re-using bricks is therefore common and allows to optimise the management of time in the creation of a new software. It is in this re-use of already existing bricks that we find the sequential nature of innovation in the software sector. According to a seminal study by the Fraunhofer Institute conducted in 2001 in Germany, the reuse of code is an important element during the development of a new software. One third of the new software contains more than fifty percent existing code.

The history of many popular software programs shows this inner property of innovation in the software sector: each new product has introduced new innovations on top of previous innovations, no software was created *ex nihilo*.

A well-known example is the development of web browsers. When in 1989 the World Wide Web technology was invented by Tim Berners-Lee at CERN (European Center of Particle Physics, based near Geneva on the French-Swiss border), rules were defined formatting pages according to the HTML language. To interpret 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 fulfilled its goal. Then HTML was enhanced to include graphical data and Mosaic came up with an intuitive interface and the ability to display graphic data. Then Netscape (Mosaic with a few new features) appeared and quickly became the leading browser. Seeing the browser as a threat to its monopoly, Microsoft reacted and launched its own browser, Internet Explorer. To develop its browser quickly, the software giant simply bought the code of Mosaic and hired lots of Netscape employees. Internet Explorer was nothing more than Mosaic-plus.

It is one lesson of this history (a repeat of the development of spreadsheets with Lotus 1-2-3 and Excel or any other kind of software) that innovation took place sequentially, each new software introducing its own new features on top of existing ones. 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 to the patent monopoly duration (which lasts twenty years).

With such an incompatibility between the cycle of innovation and the patent protection we might face a *tragedy of the anticommons* 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. There will be a large increase in the costs of transactions since each software fragment is held by a different owner. Only large companies with deep patent portfolios will be able to avoid the tragedy of the anticommons by creating patent pools between them (this may be seen as a breach of European antitrust laws). Less innovation will occur since small and innovative players won't be able to get the rights for each block included in their software for the following reasons. First, there is no compulsory licensing, therefore there is no incentive for patent owners to grant a licence to an ingenious competitor. Second, the costs in time and fees is too high compared to the average software life cycle (3 years).

3. Patents and Cycles 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 figures of the Fraunhofer Institute study: in the German market more than seventy percent of companies develop their software in less than twelve months. 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 physical innovations in traditional manufacturing industries.

The life cycle of the released product is itself very short: more than seventy five percentof customers are replacing their software each year and almost fifty percent are replacing them every six months. This shows a true dynamic, encouraged by software publishers (economists have studied this behaviour as planned obsolescence

Needless to say, software innovations are made available on the market before their creator is granted 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. In other words, if you applied for a patent, the innovation you wanted to protect will be already outdated in the market by the time the patent was granted. Competitors and the customers (in a free market) will have decided to embrace or to reject your software.

As we have seen in 2.2, software is made of hundreds or even thousands of algorithms. Companies such as IBM are filing more than five hundred software patent applications per year in the USA (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.- Any new software built upon previous innovations is likely to infringe on many patents, since many basic software elements or algorithms are already patented (Palette patent from Adobe, LZW compression algorithm patent which covers *.zip*, *.gif* and *.pdf* files, etc.).--

Many software firms do not have the time nor the necessary knowledge to browse patent databases to check if each algorithm inside their computer program may potentially infringe on a patent claim. They will have to hire patent experts. This cost constitutes not only a financial cost, but also a time-to-market cost since the R&D department (the programmers), for each new line of code, will have to ask to the patent department to look for prior art in patent databases. Once patented algorithms are identifies, each line of code will have to be negotiated with the patent department.

Another reason why software patents are dangerous for independent developers is that patent claims typically are drafted in a language incomprehensible to the average programmer. Thus the patent may not provide any useful information (the source code); and it may be written in such a broad way that other technical solutions are also infringing the claim. Therefore patenting is more and more about finding ways to describe a problem than to give the solution to the problem, making it difficult 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, software was mainly created by hardware manufacturers, scientists or academic researchers. Commercial software was non-existent and companies, particularly in the banking and insurance sectors were developing their computer applications in-house. At the end of the seventies, the first personal computers appeared. With this democratisation of the computer the first independent software publishers were founded, among them the leading players of today's sector: Microsoft, Lotus or Adobe.

As we have argued in the previous sections, the process of software creation involves only logic; no physical processes are needed. The capital required to develop and publish software is minimal; you only need a computer and enough money to pay for electricity and telecom bills. It is therefore not surprising that the world center of software innovation is Silicon Valley with liberal economic conditions and a steady supply of young graduate engineers. Students can start-up a software company taking with small financial risks.

Unlike traditional manufacturing industries, where variable costs are a major part of the total cost of the product, software production costs are near zero. In an Internet environment, even transportation and duplication costs are carried by the customer. The only remaining costs are fixed costs, that is to say the costs of designing, developing and debugging the software.

The constituents of such costs are mainly time and the training of the employees; computer cost are becoming less and less insignificant (with purchase costs falling below 1200 € per machine). In summary, in order to publish software you only need knowledge in computer science, a computer and time at hand. These criteria a met by many young educated graduates. Perhaps it is not surprising that in a 20 years old sector dominated by giant firms such as Microsoft, SAP or Oracle, the main innovations still come from start-up micro-companies. Prominent examples include:

- the web browser with Netscape
- data compression with shareware Winzip
- MP3 democratisation and skins interfaces with shareware Winamp
- P2P with Napster
- DivX video format.

Statistical surveys conducted in Europe confirm this picture, as we can see in the following graphic, created with the results of the Fraunhofer Institute study.

!!! <graph_page_7.png> !!!

As indicated in this graphic, almost 75% of the German software publishers count less than 50 employees, and only 4% employ more than 250 people. Similar results appear in France where 58% of the companies count 2 employees or less and only 1,3% count more than 100 employees. The Fraunhofer Institute's figures should be representative for the whole European software sector; Germany being the leading European software producer and largest market.

In conclusion, micro-structures appear to be the most frequent kind of firm in the software sector, entrepreneurs needing mainly human capital. Within such an industrial structure, software patents will create a non-natural barrier to entry in the software market. Small firms do not have patent departments nor the liquidity to file patents. In contrast large companies, such as IBM, have already a pipeline of thousands of granted US patents. They only will have to file the same patents in a fast track procedure with the European Patent Office.

Medium sized innovative European companies will be compelled to innovate and file for new patents, with the inevitable delay while American players will simply use their existing US patent portfolio in Europe. According to US figures, the average cost of patent litigation is about half a million Euros. European innovators may not even be able to afford the opportunity to find prior art challenging already granted patents. Small companies may simply leave the market.

The patent system appears designed for large companies because of long delays from file to grant, high fees, and long and costly trials to enforce legal rights. Moreover the value of a single patent is decreasing because many patents are now granted on obvious or already existing innovation. Companies feel compelled to file many patents on the same technology and only multinationals with deep pockets can do that. Considering these facts, software patents will be detrimental to the European software market.

5. Patents and Loss of Consumer Well-Being

A patent is a monopoly granted by the State. The owner can therefore sell products incorporating a patented invention at higher prices to amortize research and development costs. As there is no solution of compulsory licensing in the case of software patents, owners have little interest in selling a licence to competitors that are more price-competitive or innovative. This reduces the *ecology* of the market to very few players.

The price of innovation will not be decided by the market but by the patent owner. This leads to the well-known negative effects of monopoly pricing (the price is higher than a market price, and the company may take the choice to under-supply the market). All else being equal, patents have a negative impact on consumer well-being. Many customers will not have the opportunity to use innovations which in a free market system they would have bought.

6. Conclusion

The European software market is young and mainly composed of very small companies (less than 50 employees). The sector already has a great impact on the gross national product of European countries. Appreciating the power of the network effects in the software market (mainly because of interoperability issues and the need for standards) European start-ups should be nursed rather than being exposed to the anti-competitive effects of a software patent system. As the Silicon Valley model has shown, venture capital can support a flourishing software industry. To develop a venture capital market in Europe should be the main policy challenge for the coming years. Patents are monopoly rights granted by the State that lessen competition on the market, induce higher prices, slow down innovation, and encourage cartel behaviour (e.g. patent pools).

While the term of patent protection is at variance with the software innovation cycle (twenty years against three to five years) Europe should refuse software patents. In a European Commission survey about the patent needs of the software sector, 90% of respondents spoke out against software patentability. Following suggestions by American and European academics, Europe may want to create a *sui generis* protection, with a far shorter duration suited for software 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 Institute / Fraunhofer Institute (September 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>

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

Software Innovation Cycle and Patents

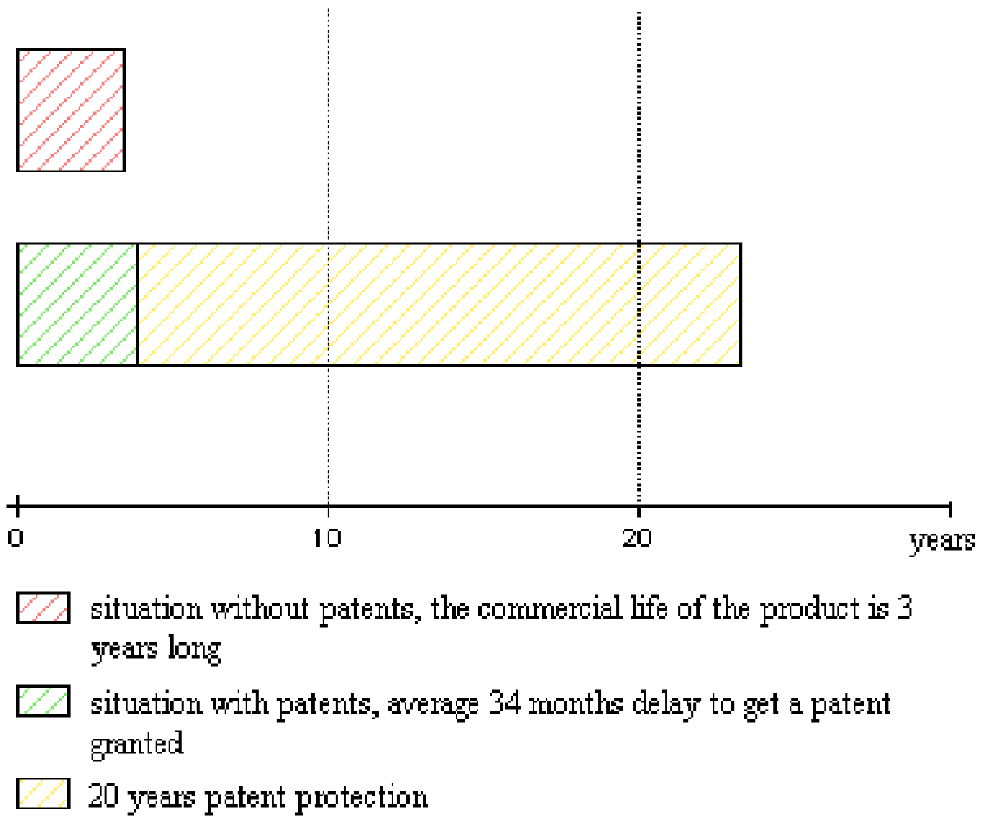

Figure 1: Incompatibility with the Software Innovation Cycle

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 is no protection for a small company.

As the patent protection does not match the life cycle of software this means that almost six generations of innovations can be blocked by the patent owner. This effect is contrary to the philosophy of giving incentives to innovators.

Impact of Software Patents on the Development Process

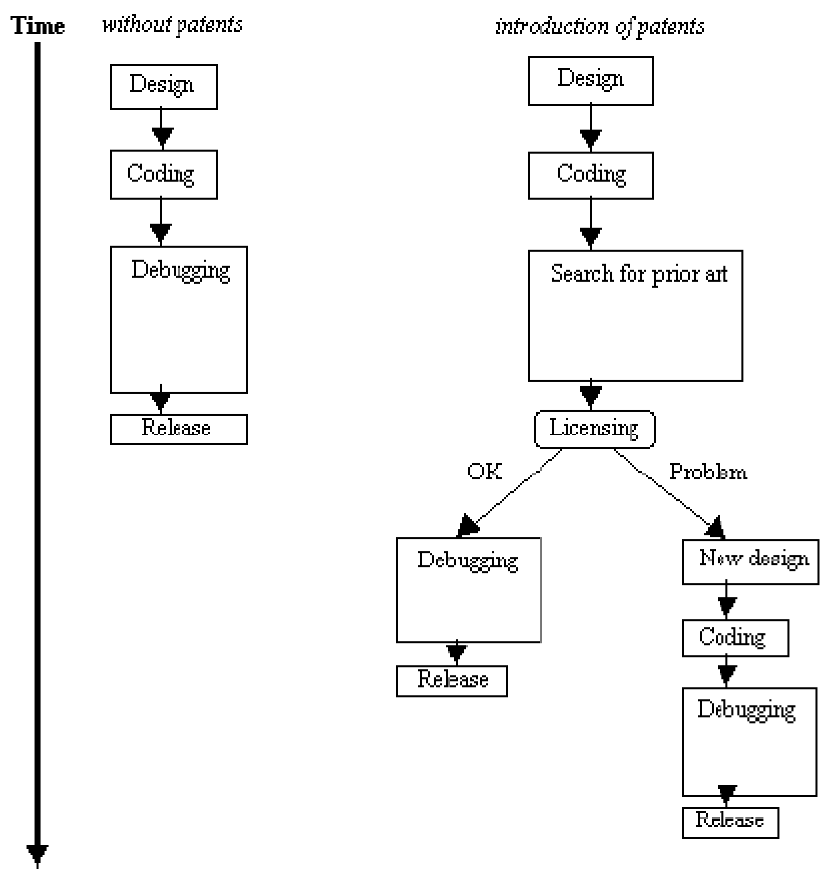

Figure 2: Patents and Slowdown of Software Innovation

We see on this synthetic graphic the differences between the cases of publishing software without software patents and with software patents.
To be as simple as possible we have not included the impossibility to circumvent the patent(s) with a new design (that would mean that the project has to be dropped); and we have summarized the licensing discussions to a very short period of time.