Software patents and their impact in Europe

Joaquín Seoane Pascual
Professor
Department of Telematic Systems Engineering
Universidad Politécnica de Madrid
jseoane@dit.upm.es

Ramón García Fernández
Researcher
Departament of Chemical Physics I
Universidad Complutense de Madrid
ramon@jl1.quim.ucm.es

December 2000

Contents

1 Introduction

At the moment, a major debate is taking place in Europe regarding software patents. We have prepared this report to relate our knowledge and points of view on this topic.

A good proportion of the current debate on software patents (with arguments both in favour and against) is hypothetical, that is, on how the software patents *should* affect technological development. These arguments do not take into account the true nature of the sofware industry. This report is based on *real facts*, obtained from the vast experience on this issue that has been accumalated in the United States, where the software has been patentable for a number of years.

Firstly, we shall demonstrate that the success of a software company is in its capability to develop quality products based on already known ideas, rather than conceiving and putting into practice new ideas. In order to do that, we include a testimony from Joel Spolsky, who was program manager for Excel 5.0, who gives us his view on the reasons for the **success** of companies such as Microsoft and Oracle.

Secondly, we shall explain how a complex software program is mainly built from **numerous basic components**, most of which pose problems that a competent programmer is able to solve within a short period of time, thus they may be considered trivial. These solutions are found over and over in what may be classed as 'similar' problems. Nevertheless, if these solutions were susceptible to be patented, the programmer's work would be made tremendously difficult, due to the need of verifying whether every single solution is already patented, negotiate the right to use the patent or avoid it, looking for an alternative solution, possibly more complicated and inefficient.

We shall then analyze why a significant number of patents are granted for solutions that are very **simple**, from a programmer's perspective. We shall see how a good laywer can make an obvious invention seem complicated and original, and real examples on how both a Patent Office and a Court consider the proposed method patentable.

We will also comment on other type of patents, generally very simple but potentially very harmful; those that apply to the **general idea of a program**, as most of the time they prevent others from solving the same problem, which is against the free competition and the incremental evolution of the characteristics of software applications.

Even though the most important problem we face is the granting of patents for very simple methods or ideas, serious problems also arise when they are granted to more complex inventions. The most serious ones are those whose purpose is to prevent the creation of **compatible programs**. We shall explain the most frequent types and *why there are very dangerous for the European software industry*.

Frequently, patents are granted on **user interfaces**, that is, on the way that a user interacts with a program. There are severe competition problems related to the intelectual property of user interfaces. Due to this fact they were excluded from the protection of copyright in the United States and other European countries. Nevertheless, patenting user interfaces is being allowed.

We will also see why patents are useful as threats. *A patent trial is so expensive that it is more profitable to pay for an invalid patent that to go to trial (even when one wins).* For this reason, lot of companies obtain patents with the sole aim of defending against other patents. These are known as **defensive patents**.

We will finish this report with some conclusions on the adverse consequences of software patentability in the industry in general and in Europe in particular.

2 Microsoft success: “convert capital into software that works”

Joel Spolsky was Program Manager of the Microsoft Excel spreadsheet between 1991 and 1994. He was responsible for version 5.0, the first Excel version that surpassed others in sales. He created the idea of “Visual Basic for Applications”. In this essay he explains what he believes are the most important reasons for the success of this company, having nothing to do with patents[1].

> ***I’m convinced that most people think about software companies in an upside-down way****. The common belief is that when you’re building a software company, the goal is to find a neat idea that solves some problem which hasn’t been solved before, implement it, and make a fortune. We’ll call this the build-a-better-mousetrap belief. But the real goal for software companies should be converting capital into software that works. If you understand this, it’s easier to make the right strategic decisions.*
>
> *The trouble with build-a-better-mousetrap is that there’s not a lot of evidence that it works. First of all, many of the most successful software companies (Microsoft and Oracle, for example) don’t really "innovate" in the sense that they are not really solving problems that haven’t been solved before. In any market, it is exceedingly rare that you get to keep your invention to yourself. Everybody has competition. Wall Street Weenies and lawyers starting high tech companies tend to think they can protect themselves from this with patent protection. Ha! I can hardly think of a single case of a company successfully protecting themselves from competitors because of a patent. (Stac is the only case I can think of, and where the heck are they?)*
>
> *The next problem with build-a-better-mousetrap is that we’ve reached a state with Internet software where there is too much money around chasing the same lame ideas. Call it the idrive-xdrive-swapdrive-freedrive (free hard disk drive on the Internet in exchange for advertising) phenomenon: suddenly thirty-seven companies pop up offering exactly the same service for free. There are a zillion examples of this. Petshops-on-the-net. Urban-video-rental-delivery. Cosmetics websites. When this happens, the business challenge switches from being a technical challenge that needs good programmers to being a marketing challenge that requires, somehow, the ability to break through the pack and establish a brand name, something that is vanishingly improbable. Not only that, but VC money is impatient. That means that investments which take a really long time to develop won’t get funded, which is why anything really interesting or hard to copy won’t even get funded. One reason that there are over thirty companies whose entire goal is providing free hard drive space on the Internet is that writing the code for such a service is so easy.*
>
> *There’s a different way to think of software development. Imagine that the goal of your software company is* ***not*** *to solve some specific problem, but to be able to* ***convert money to code through programmers****. That’s a little bit strange, but bear with me. A software company has to think of recruiting the right people as its* ***number one problem****. If you are successful, this can solve any other problem. Hire smart people, and they will produce good stuff that you can sell and make money off. Then everything else follows. Microsoft has the ability to crush its competitors because it has the ability to deploy so many programmers. When Microsoft released Internet Explorer 3.0, fast on the heels of IE 2.0, it was shocking just how good a job they had done. Not only did the replicate every feature in Netscape’s browser, but they added some more features too, and did it all with an architecture that was robust and strategic. While it*

[1] This essay is ©Joel Spolsky, 2000. Included and adapted with authorization. The original version can be found at http://joel.editthispage.com/stories/storyReader$17. Joel Spolsky’s e-mail address is spolsky@panix.com

is true that Microsoft used its operating system to help push its browser, it is also true that they just wouldn't have gotten away with this if their browser wasn't great. (Case in point: even though Windows out of the box can play MP3 files, everyone I know uses WinAmp, not the Windows Media Player, to listen to them. Even though MSN is on the desktop, everybody uses AOL. Back when the browser integrated with Windows was crap, Netscape had 80% market share.)

[. . .]

If you want to be the number one restaurant in town, you have to worry about how to get the best chefs and the best ingredients. You are a factory that converts raw ingredients and chefs to a dining experience. If you are a movie studio, you have to worry about getting the best actors, directors, and writers. You are a factory that converts talent into entertainment. So if you are a software company, you have to get the best chefs and directors: you are company that converts talent into code. The right talent knows how to make the right code which will make you successful.

3 Software development

The software development process is composed of several stages. Once the problem to be solved and the general idea on how to do it are identified, its functionality is specified in detail. After that, the actual programming work starts; splitting the computational problem into several small and interrelated sections. Programmers write the code for these small parts, testing them and debugging the errors separately. Finally, all the pieces are integrated and tested as a whole. Once the integration tests have been passed, the program is installed or distributed, starting the maintenance stage, in which the errors discovered by the users are fixed, modifying the functionality according to their needs and making the system evolve. This phase is the longest and most expensive, and the one in which the product quality and the software company proficiency become aparent.

So, we can see that programming consists of breaking down a problem into several parts, each of them being converted into a sequence of programming instructions. Every part has very little value when taken in isolation, and only the combination of all of them, the whole program, is useful to the consumer.

For each of the parts into which the program was divided, the programming solutions may prove either simple or complex. If a solution is complex, it may be split again into a number of simpler parts. If it is easy, it can be solved using common sense and the resources every professional programmer should have. If a solution is difficult, it can demand a research activity in order to design a suitable algorithm[2].

The vast majority of the small problems solved by a programmer belong to the **easy** category, if he or she is sufficiently qualified. The ability to find a combination of computer instructions to achieve a certain objective is essential in a good programmer. Here one has a proof: *in the ACM Collegiate Programming Contest, the participants must solve difficult problems in 6 hours*[3]. We suggest you to compare a software patent, for example the USA patent 4197590 on the use of the

[2]Although current laws do not alow patenting *algorithms*, in practice one can. As most algorithms have a practical function, they are also *methods* and, therefore, are suitable to be patented

[3]ACM (Association for Computer Machinery) is a worldwide organization of computer professionals and students. This contest is on talent and speed in problem solving. 8 problems are proposed, to be solved in 6 hours. Usually, no participant is able to finish the 8 problems, but it is usual to solve at least 4. The contest web page is located at http://acm.baylor.edu/acmicpc/.

logic operation *or exclusive* to draw cursors (also granted by the patent offices in France and United Kingdom) with any of the problems proposed in last years world finals of the ACM Programming Contest[4].

Moreover, and indepently of how easy of difficult the small problems are, they usually have a single solution or a reduced set of reasonable solutions. If the solutions to these small problems were allowed to be patented, the **accidental infringement** of a software patent would be highly probable. Also, it would be easier for a programmer to solve a problem on his or her own than having to search through a patent database to see whether the problem has already been solved by any valid or no longer valid patent. In case that it is valid, he would also have the additional task of negotiating the license or to try to avoid it by looking for a different solution.

The high probability of accidental patent infringement may force sofware companies to go to greater lengths hide their source code, as open source would make suing them easier. This has severe consequences for security and maintenance. For security because life and public health depend on certain applications, and those making use of them cannot allow the risk of not knowing exactly how they work. On maintenance as most times a client needs to modify a program to fulfil a requirement that was not anticipated by the program designer.

In conclusion. *The granting of a monopoly lasting 20 years for a method to solve a problem that requires a only few hours to be solved, or for an idea that appears in a moment of inspiration that can happen to anyone, does not look to us beneficial for society*.

Up untill now we have referred to patents affecting small and simple parts of a program, but the same can be applied to others programs, such as the patents on user interfaces and the global idea of a program, which we will analyze in specific sections of this document.

4 The problem of obvious patents

One of the more criticized characteristics of the patent system is that patents are granted to very simple, even trivial, inventions.

Defenders of software patents attribute the problem to the difficulty that the evaluators have in finding whether there exist previous inventions. We do not agree. *The problem is that law only demands that the patented method "be new, a technical advancement, and inventive"*. The work of a few hours needed to solve a typical computational problem may fulfill this criterion, although it should not be allowed to be patented for the reasons we outlined above. Moreover, as we shall see later, *the interpretation of the law made by both the United States and European Patent Offices is that any method, no matter how simple it is, should be patentable.*

Given the technological complexity of programming, it is difficult for non-programmers to decide whether a method is simple or not. *We believe this the reason why a number of lawyers and politicians are in favour of software patents.* For example: in the United States, when the Patent and Trademark Office started to grant obvious patents on business methods, worry began to grow; right now there is a discussion going on in that country's Congress in order to restrict patents on business methods.

Nevertheless, that Office has been granting obvious software patents for a long time and there has been almost no politician who has complained[5]. The reason is that anyone can understand that a patent on a business method is trivial, but only a programmer can understand a software patent.

[4] Available on the Internet at http://acm.baylor.edu/Past/PastProblems.html.

[5] Although there have been numerous complaints from programmers and companies.

This problem has been know for quite some time (there was a public consultation in 1994 and the problem still persists). Here you are an excerpt from a Red Herring magazine article[6]:

> *John Barton, a professor of law at Stanford University, proposes a fix. He advocates raising by a factor of ten the "new and nonobvious" standard required for a patent. This would result, he says, in fewer "nuisance patents," and it might benefit overall innovation. "It would reduce the number of patents that large companies can assert against smaller ones," he says. "Big companies using large patent portfolios to get payments from others are becoming a serious problem in the IT community."*
>
> *(Red Herring, May 1999, by Luc Hatlestad)*

The lack of rigour demanded by the United States Patent and Trademark Office is a very well known fact. Thus, the legal offices (that are usually in favor of software patents) do not doubt in using it. Next, we include an excerpt of the instructions a lawyer's office wrote for its clients[7]:

> *(We have put in bold the part relevant to this discussion)*
>
> *[Question] What should I consider in deciding whether to apply for a patent?*
>
> *[Answer] First, consider whether the invention provides a commercial advantage. If so, patent protection may be necessary to protect your investment.* ***Don't belittle your invention. Although an invention may seem straightforward to you, it still may be patentable. You should avoid self-censoring****. You could not only lose valuable rights, but a competitor might obtain a patent on your invention. In such a case, you would incur substantial expense defending against it.*

The problem is so serious that there are companies, usually known as litigation companies, whose only activity is patenting very simple ideas so that when somebody has the same idea they demand renumeration.

4.1 History of the problem of obvious patents

The problem of the obvious patents was considered last century in the U.S.A. At that time the U.S.A. Patent Office acted as now, allowing one to patent any new idea, even if it was a very simple idea. For example, in 1867 The United States Patent and Trademark Office (USPTO) granted a patent to "A hotel registry book with the margin of its leaves occupied by advertisements"[8]. Last century businesses used these patents to dominate markets and to eliminate competitors. An example of it were the "Big Telegraph Wars" in the 1870's.

In the 1930's, the United States Government changed its policy and began to see the patents as a monopoly mechanism. With the only exception being the pharmaceutical industry, the industry began to restrict the use of patents to solely defensive aims.

But at the end of this century, the previous situation returned. In 1982 a new court was created, the "Court of Appeals of the Federal Circuit" (CAFC), with twelve judges in favour of allowing more patents. Before the CAFC creation, 75% of the lawsuits that included patents were dismissed. Today nearly 72% of the lawsuits on patents are accepted, and the indemnities by infraction are

[6]The full article can be found at http://www.redherring.com/mag/issue66/news-sue.html.

[7]The office is Wolf, Greenfield & Sacks. These instructions can be found in their web page at http://www.wgslaw.com/news/faq/qa-soft.html.

[8]We have found this patent in the testimony of the director of the USPTO during a debate. It can be found on the Internet at http://www.oreillynet.com/pub/a/patents/2000/10/23/isoc.html?page=2

substantially higher than before. In addition to that, it changed the doctrine of the Supreme Court of the United States, who had seen the patents as anticompetitive and, in spite of that, since then it has been promoting the maximum power for the intellectual property. All this has relaxed the requirements for a patent to be valid in court.

Lately, there are increasing movements towards this direction and courts are accepting pattents very close to the ridiculous. *It is clear that granting a large number of low quality patents is not due to the difficulty in finding previous inventions. It is a political decision*.

4.2 The case of the European Patent Office (EPO)

The philosophy of the European Patent Office is similar to the one in the United States: every original idea, even if it is very simple, must be suitable to be patented. Let us see, for example, patent EPO 926584:

> System for controlling the use of a software item
>
> *Abstract: A system is described for controlling the use of a software item, such as a program or data, in a computer network, comprising at least one server and a plurality of local computers, each of the local computers having means for downloading an item from a server to the local computer. The system comprises means for controlling access to the downloaded item, so that the provider of the item keeps control over a downloaded item.*
>
> *Claims*
>
> *1. System form controlling the use of a software item, such as program of data, in a computer network, comprising at least of one server and a plurality of local computers, each of said local computers having means for downloading an item form a server to the local computer, said system comprising means for controlling access to the downloaded item.*
>
> *[...] (The rest of the claims are not relevant for this discussion. Let us remember that each chaim is an independent invention that can be used in a lawsuit).*

Let us take a more detailed look at this patent. In the summary we have a description on how a server on the Internet works: a central computer (usually belonging to a company offering a service) and "local" computers belonging to clients and conected in order to obtain this service.

The only new fact that can be found both in the summary and in section 1 is that "it would be useful to have some mechanism to restrict what the user does with the downloaded sofware", but we are not even told how is this mechanism. The defect of this patent is not the absence of any new idea, but the fact that is has *nothing* (in other sections there are proposals for access control, but these sections are independent "inventions" in case of a trial). *A patent has been granted to an idea that anybody having such a need could have found*.

The European Patent Office has also accepted the Amazon 1-Click patent, that will be discussed later, in section **??**.

4.3 Making an obvious patent look non obvious

A good patent lawyer can make an obvious idea look a complex method. In order to do so, *the only possible way to use the idea is described with a lot of detail and with a complex technical language*. This is better explained using an example: the United States Patent 5963916[9].

[9] This example is debt to Richard Stallman.

This patent's "method" is that "it's useful for a client to listen a piece of a song to decide whether he should buy it". We apply this idea to an Internet music shop. Describing in quite some detail how a web server usually works (applied to this particular case) we convert this idea into a sophisticated method. Let us see the patent:

> *1. A method for enabling a remote user to preview a portion of a pre-recorded music product from a network web site containing pre-selected portions of different pre-recorded music products, using a computer, a computer display and a telecommunications link between the remote user's computer and the network web site, the method comprising the steps of:*
>
> *a) using the remote user's computer to establish a telecommunications link to the network web site wherein the network web site comprises (i) a central host server coupled to a communications network for retrieving and transmitting the pre-selected portion of the pre-recorded music product upon request by a remote user and (ii) a central storage device for storing pre-selected portions of a plurality of different pre-recorded music products;*
>
> *b) transmitting user identification data from the remote user's computer to the central host server thereby allowing the central host server to identify and track the user's progress through the network web site;*
>
> *c) choosing at least one pre-selected portion of the pre-recorded music products from the central host server;*
>
> *d) receiving the chosen pre-selected portion of the pre-recorded products; and*
>
> *e) interactively previewing the received chosen pre-selected portion of the pre-recorded music product.*

Let us know have a more detailed look at this patent.

> *1. A method for enabling a remote user to preview a portion of a pre-recorded music product from a network web site containing pre-selected portions of different pre-recorded music products, using a computer, a computer display and a telecommunications link between the remote user's computer and the network web site*

The client needs a computer to be connected to the Internet. Usually, personal computers include a screen and both the music seller and the client are connected to the Internet, acting as a "telecommunications link".

> *a) using the remote user's computer to establish a telecommunications link to the network web site*

The standard method to use the web nowadays consist of a browser on the client computer that connects ("establish a telecommunications link") with a web server.

> *wherein the network web site comprises (i) a central host server coupled to a communications network for retrieving and transmitting the pre-selected portion of the pre-recorded music product upon request by a remote user and (ii) a central storage device for storing pre-selected portions of a plurality of different pre-recorded music products;*

Here we are informed that the web site is a computer (actually, all of them are). As this computer is waiting for client requests, it is a server (in the text, a "central host server" sounds much more technical that simply a server). It has a hard disk ("central storage device") and an Internet conection. It also reminds us that the computer function is sending and receiving data (the network conections are used for sending and receiving data). The hard disk is used for its usual function: storing data, music in this case.

> *b) transmitting user identification data from the remote user's computer to the central host server thereby allowing the central host server to identify and track the user's progress through the network web site;*

This is a conventional technique in web servers, known as "cookies". Cookies are a mechanism used to identify users (for example, in order to have each user downloading a web page with different preferences). All that is said in this paragraph is that we will "use cookies".

> *c) choosing at least one pre-selected portion of the pre-recorded music products from the central host server;*

Before the user listen to the music fragment, one must choose a song.

> *d) receiving the chosen pre-selected portion of the pre-recorded products; and*

When the user clicks on a link in a web page, his computer receives the selected content. This is the way in which the web works.

> *e) interactively previewing the received chosen pre-selected portion of the pre-recorded music product.*

Finally the user listens to the music. This happens automatically with most web browsers, when the user clicks on a link to a sound file.

With the technique used by this patent, it is possible to patent any simple idea.

4.4 Mixed cases: the Amazon 1-Click

There are many intermediate degrees of obvious patents. This is to say the case of patents consisting of an obvious part, and a subsequent part that is not obvious. However, the non-obvious part cannot be patented. Its purpose is avoiding the restrinctions on patentability. For example, using this strategy you can patent ideas that, on their own, are not patentable.

A good example is the famous Amazon 1-Click patent, granted by the United States and European Patent Offices (US 5960411, EPO 902381). The patent covers the use of Internet "cookies" in order for the user not being required to write his address and credit card every time he buys an item. This idea es evident, as it was one of the aims in mind when the "cookies" system was created (part of the HTTP standard, that is patent free), and so is acknowledged by the Amazon CEO, as we will see next.

In spite of that, there is something original in the 1-Click system. The user interface to buy books in Amazon allow to request a book with a single click on a button. Till that time, the Internet shops used the metaphor of the shopping cart. We can confirm this reading what this company's CEO said[10]:

[10]The cite is from Tim O'Reilly, that held a telephone conversation with Jeff Bezos, due to the polemic Amazon patent. Tim's text can be found at http://www.oreilly.com/ask_tim/bezos_0300.html

> *First off, Jeff wanted to explain why he thought 1-click was original enough to patent. It has nothing to do with the implementation, which he admits is fairly trivial to duplicate, but with the reframing of the problem. At the time he came up with 1-click shopping, everyone was locked in to the shopping cart metaphor, because that is what you do in the real world. You pick up an item and take it to the counter to buy it. On the Web, he realized, something very different was possible: all you had to do was point to an article, and it was yours.*

It is true that Amazon has been the first bookshop in the Internet that has taken care to make its service easy to use. But *applying user interface design techniques, that exist from the 1980's, to the Internet, does not make them inventors*[11] .

5 Patents on the general idea of an application

These patents forbid the development of a program that has a concrete objective. The defenders of this kind of patents say that anyone having an idea for a program must get a reward for it. But *the idea on a application usually comes from observing the users or obtaining their requests*. Due to this fact, our opinion is that these patents do not imply enough work to be worth it.

A typical (and funny) example is the US Patent 6049811, that covers the idea of a program to fill in patent applications. The idea came from observing the possible users, as we can deduce from this excerpt of a New York Times article[12]:

> *[...] many inventors who submit their own applications end up turning to a patent attorney or agent after their documents are rejected as inadequate. That requires the inventor to explain his invention to the attorney so the application can be made as legally binding as possible. And the lawyer's expertise costs money.*
>
> *James D. Petruzzi says inventors can avoid some of that trouble and expense with his software. Petruzzi is a patent attorney himself – but he does not believe his invention will harm those in the business.*
>
> *"We found in our practice that a lot of people do not have the resources to pay a patent attorney to go through the process, and they are being cut out," Petruzzi said. "This software enables a lot of people to get into the system and apply for patents."*
>
> *(New York Times, "Product Designed to Ease Patent Process Wins Patent", by Sabra Chartrand, May 1 2000)*

From the last paragraph it is clear that the idea came from observing his clieants while he was in practise as a lawyer.

Generally, the characteristics of computer programs **evolve** according to an **incremental** process. As the users acquire experience with a program, they notice that they miss certain characteristics. Salesmen take this into account and add them to the programs. If it were possible to patent the general idea of a program, this evolution would only be possible if the patent owner were willing to allow it, discarding any possible competition.

[11] For example, the first edition of the book "Designing the User Interface" by B. Shneiderman, published in 1987, explains how to design an application user interface starting from the functionality. Using this book's method, the 1-Click idea is evident.

[12] The article is available on the Internet at http://partners.nytimes.com/library/tech/00/05/biztech/articles/01pate.html (you need to sign in (at no cost) to have access to the article)

Typical examples of program evolution can be found in word processors (WordStar, then WordPerfect and Microsoft Word) or spreadsheets (Visicalc, then Lotus 123, then Excel). For example, *Visicalc* was the first modern spreadsheet. According to their authors, it is based on previous column-based business data processing programs. The program *123* from Lotus was the next spreasheet. Lotus observed that users wasted a lot of time repeating long sequences of operations. So, they added *macros*, a method to repeat a sequence of operations in a comfortable way. Later, Microsoft observed that the users were using the spreadsheets to store lists of data (till then it was believed that the main use was in finances). So, they added characteristics such as easily sorting data or automatic data input. Microsoft also added "Visual Basic for Applications", as an alternative to macros. Later, Lotus added all these characteristics to its spreadsheet.

This leads us to another argument: the **converging evolution**. If two applications are designed to solve the same problem, they will have to end up having the same characteristics. Although the imitation does not look a good thing, it is logical. Due to this, if we have a look at the latest versions of spreadsheets or word processors, we will see that they have very similar functions, even though the menus or icons are different. If it were possible to patent the improvements, the converging evolution would be impossible.

6 Patents on non-obvious methods

Not only the patents on very simple ideas or methods are dangerous. Nowadays there are million of programmers working to solve non-trivial problems. So, the probability of discovering solutions to non-obvious problems in different places almost at the same time is very high. Although in some cases the investment needed to reach these solutions may be high, it is usually much smaller than in other industries, and its patentability only rewards those having the luck to register it first, with the aggravating circumstance that the investment needed to put a sofware idea into production is almost null.

As an example of simultaneous inventions we can cite the compression algorithms invented by Ross Williams and published in April 1991. They could not benefit society due to the US patent 5049881, requested some months before and granted afterwards. In this case the patent did not contribute to the publication of the method, as it was done anyway.

The patentability of software adversely affect the so called *free software*, as it would be obliged to look for non-patented solutions for all and every functions, given its inability to collect money for paying the licenses, even if they were very cheap, and it is absolute inability to hide its use, as the source code is open.

We will describe next two types of patents that can be non-obvious and tend to favour monopolies, as they prevent from making compatible programs.

7 Patents against compatibility

We call that way those patents designed to make impossible to create a program being compatible with another one, as any compatible program infringe these patents. These patents are usually used by dominant companies to tie their clients. Usually the European sofware companies are not dominant, so that these patents are specially dangerous for Europe.

Generally, these patents affect a "language" used by two computer programs to communicate with each other, or to store data that can be read by other program. Frequently there is no special

advantage when using one language or the other. But, above all, these patents *are not beneficial for society* as they make competition impossible.

We believe that these patents should be forbidden due to this reason and to be coherent with the spirit of the European Community directive 91/250/EEC on 14th of May 1991 on computer programs copyright. This directive allows reverse engineering (normally forbidden) of a program when it is necessary in order to achieve compatibility. This would be contradictory to allow restrictions to compatibility with patents, when the copyright laws are designed to avoid these restrictions. Moreover, the experience of allowing reverse engineering for compatibility reasons has been positive. Using reverse engineering[13] a Microsoft Windows NT compatible file server called "samba" has been developed. This program was faster then the Windows NT server itself, and this fact obliged Microsoft to improve its speed. The Service Pack 4 version 4.0 achieved a noticeable increase in speed, and this fact was used by Microsoft in its advertising. So, the reverse engineering has brought competition that, in turn, has been benefitial to users.

These patents can be classified according to the kind of compatibility that they are meant to avoid:

- **File formats**. These patents cover the format used by an application to read and write data from files. For example, the word processors write documents to files and the users exchange them. If the manufacturer of the most popular word processor patented the format of its files, all the users would be obliged to buy these program, in order to be able to exchange documents among them. If, in addition to that, the word processor only works in a certain operating system, made by the same company, the users would also be obliged to use this operating system.

  For example, the Microsoft ASF format, used for audio and video, is patented (US Patent 6041345). ASF is not only used to store sound and video in a hard disk, but also for transmission in the Internet (for this reason, this patent belongs also to the network protocols group). Thanks to this patent it is not possible to develop programs able to read ASF files or transmissions in platforms different from Microsoft Windows (in this operating system, Microsoft supply libraries to access the contents of ASF files). This prevents one from adapting applications written for Windows to other operating systems.

  As a proof that this is actually Microsoft's strategy, we include an excerpt from the antitrust trial judgement agains the Microsoft company. In it, the judge recognised Microsoft's desire that multimedia applications are tied to their operating system, and for this reason they attacked RealNetworks , a company offering operating system independent alternative services, for multimedia applications[14]:

  > *Paragraph of the Court's Finding of Facts (we have put in bold the part we feel to be relevant):*
  >
  > *111. RealNetworks is the leader, in terms of usage share, in software that supports the "streaming" of audio and video content from the Web. RealNetworks' streaming software presents a set of APIs that competes for developer attention with APIs exposed by the streaming technologies in Microsoft's DirectX. Like Apple, RealNetworks has developed versions of its software for multiple operating systems. In 1997, senior Microsoft executives viewed RealNetworks' streaming software with*

[13]The network protocol is needed for compatibility. Most of this protocol is public and it is not necessary using reverse engineering. But an important part of it, the "NT domains", has been kept in secret by Microsoft, to oblige users of networks with Windows 95, 98 or NT machines, to use a Windows NT server.

[14]The full document can be seen in the Internet at http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm.

*the same apprehension as they viewed Apple's playback software – **as competitive technology that could develop into part of a middleware layer that could, in turn, become broad and widespread enough to weaken the applications barrier to entry**.*

- **Network protocols.** A network protocol is a language used to communicate two programs across a network. Usually, the purpose of these patents is to oblige the users of the dominant program to buy another program manufactured by the same company.

  A curious example of a patented network protocol is WAP (US Patent 5809415), which is used for Internet access in mobile phones. In this case, the patent owner kept in secret its ownership while convincing mobile phones manufacturers to adopt it as a standard. Now, mobile phones manufacturers are obliged to pay arbitrary fees for a protocol whose only value is that it is used by everybody.

- **Processor Instruction Codes.** The purpose of these patents is tying the applications developed for a processor to this one, preventing other processors to execute those applications.

  The processor is the central part of any computer. The processor executes applications. When a program is executed in a computer, the processors reads and interprets this application's instructions. It has commands that are executed by the processor. These commands are known as "instruction codes". In order for a processor to execute applications writen for other processors, it is necessary that it is able to understand all the instruction codes of these other processors.

  In order for a processor to be successful, there must be applications that can run on top of it. For example, Apple computers, even though they are easy to use, have fewer sales than PC compatible computers, as there are fewer applications running on the former[15]. Intel is patenting the instructions of its new IA-64 processor. That way it will guarantee that it prevents the manufacturing of compatible processors. Here we reprint an excerpt of a comment which appeared in the EETimes magazine[16]:

  > *[. . .] Intel is trying to patent the functions carried out by specific instructions.*
  >
  > *In doing so, the company appears to be, in effect, trying to patent the IA-64 instruction set itself.*
  >
  > *[. . .] "I don't think that they [Intel] are going to license these patents," said Rich Belgard, a microprocessor consultant in Saratoga, Calif. "I think they want to protect the IA-64 from cloning."*

- **Application program interfaces (APIs).** They define the language used by applications to use services not performed by themselves. An application uses the processor to make computations, but other kind of operations, such as opening a file, creating a window, or drawing in the screen, are performed asking them to the operating system or othe independent services. APIs are used to request such services.

  The *purpose* of these patents is exactly the same than the one of the patents on processor instruction codes: *tying the applications to a certain operating system*. An operating system

[15] An excellent description of the importance of compatibility among processors and operating systems can be found in the Internet address http://joel.editthispage.com/stories/storyReader$117.

[16] The full original article can be found in the Internet address http://www.eet.com/story/industry/systems_and_software_news/OEG20001027S0028.

is only useful if there are applications running on it[17]15. So, a new operating system should be able to run the applications of the other existing operating systems and, in order to do so, it must support their interfaces. For example, Microsoft MS-DOS version 1.0 implemented the CP/M interfaces, the dominant operating system at that time. This allowed that Wordstar, one of the most popular applications at that moment, worked on MS-DOS. And MS-DOS was very successful. On the contrary, the modern operating system BeOS, although excellent and *innovator* from a technical point of view, has not been successful because it's not Windows 98 compatible, and thus there are not many applications running on it.

Let us see and example. Microsoft owns the United States Patent 6101510 on the idea of using a browser as "control"[18]. This is an obvious patent (being a combination of two already existing concepts: a control and a browser). Its purpose is that the applications using the Microsoft Internet Explorer ActiveX controls are tied to the Windows operating system. Microsoft's desire of not having operating system independent APIs for Internet was the most important reason to promote Internet Explorer, as well as the reasons that took it to the antitrust trial. Here we show an excerpt of the sentence:

> *He [Bill Gates] warned his colleagues within Microsoft that Netscape was "pursuing a multi-platform strategy where they move the key API into the client to commoditize the underlying operating system."* [19]

8 User Interfaces

The user interface is the way in which the user interacts with a program. It is composed of different dialog elements (menus, buttons, windows, etc.) that are arranged in a certain way and used with a consistent logic.

The restrictions on the copy of user interfaces tie the users to the product they know. Learning to use a program is a costly and difficult task. There are a lot of proofs: you can just take a look in any computer bookshop and see the huge number of books dealing with learning how to use computer programs, or the number of courses paid for by companies. So, users try to avoid changing from one program to another which has a different external look and feel. This is an example on how a good principle, the copy protection to promote innovation, turns to be harmful if it is applied blindly.

A good example of this idea is the trial "Lotus against Borland" in the United States. Lotus sued Borland because the program Borland Quattro Pro had the possibility of using the same menu structure than the one in Lotus 123. The reason why Borland copied the Lotus menu structure was not lack of talent nor saving work, as Quattro Pro included also an original menu structure. It is clear enough that the only reason to offer the Lotus 123 menu structure was helping users having experience with this program. For this reason, Borland won the trial in the United States Supreme Court. For the same reason, several countries have laws excluding user interfaces from copyright.

Although user interfaces are not protected by copyright laws, they can be patented, and this is one of the reasons used in favour of software patents. This proves the point of view, used frequently by legal media, of defending the maximum protection without reflecting on the efect it may have for competition:

[18] A control is a user interface element that is included in the application windows. A window system has a set of standard controls, such as buttons, dialog boxes, etc., and the instructions to include them are always the same in a windows system, so that every program has a consistent look and feel.

[19] The idea is that with HTML, javascript and plugins, it is posible to develop applications that are completely independent of the operating system.

Why are so many attorneys and clients reluctant to realize or properly react to this important IP development? One reason is the enormous difference in legal costs between preparing copyright applications and patent applications; spending more money on legal fees is always a difficult subject, even if the downside is potentially much more expensive.

Yet through decisions such as Computer Associates v. Altai[20]*and Lotus v. Borland, the courts have clearly demonstrated that copyright law now provides little protection for software. Also, since copyright law, unlike patent law, can vary among circuits, more uncertainty exists with respect to copyright law.*

(National Law Journal, Mondey, April 20, 1998)[21]

Our opinion is that the lessons learnt from the user interface copyright issues must be applied to patents. *The patents on user interfaces tie users to the programs they know, that is, to the dominant companies (which are normally non-European).*

The most extreme case of user interface copy is that of **cloning**. A clone of a program is another one with the same functionality and the same user interface, but with different code. Software cloning has been used as an argument to defend software patentability.

Pure clonation is fairly rare. It creates the perception of a *copied* program and damages the public image of the sofware company. We do not know any cloning case among the main sofware companies (Microsoft, Borland, Corel, Lotus). Nevertheless, as we have shown above, it is frequent that different programs designed in order to solve the same problem end up having the same capacities.

9 Patents as threats

Frequently, patents are used as threats. Patent trials are so expensive that it is cheaper to pay for a patent that the defendant does not consider valid, than demonstrating that the patent is not valid in a trial.

Defending a sofware patent trial in the United States usually costs around half a million dollars (we do not have data for European cases due to the very small number of software patents granted). This article in the Wired magazine[22] explains these problems very well:

[...]

According to Stanford University Professor John Barton, patent infringement suits are among the most expensive kind of litigation in the US today, with the average cost of a patent suit being US$500,000 per side per claim. Not surprisingly, the cost of insurance to protect companies against patent infringement is equally steep: $50,000 per product with a $50,000 deductible in the case of multimedia software, says Rob Lippincott, president of the Interactive Multimedia Association, a trade organization for large and small multimedia publishers. "These kinds of numbers are basically intolerable", says Lippincott, adding that the cost of merely defending an infringement will wipe out most small software houses, whether they win or lose.

[20] In the Computer Associates (CA) case agains Altai, Altai used unknowingly source code stolen from CA. As soon at it got to know it, Altai destroyed the stolen source code and rewrote the corresponding program. The Court considered that the behaviour of Altai was right.

[21] The full article can be found in the Internet at http://www.ljx.com/practice/computer/0420softpat.html.

[22] The full article can be found at http://www.wired.com/wired/archive/2.07/patents.html?topic=&topic_set=.

"Patently absurd, part I", Wired Magazine, July 1994, written by Simson Garfinkel

The high cost of trials can bankrupt a small company, even if it is in the right. Here one has another excerpt from the same Wired article:

> *One person who found himself on the wrong side of such an infringement suit is Vern Blanchard, a programmer whose company was destroyed by a patent that wasn't even valid.*
>
> *Blanchard is president of American Multi-Systems, a San Diego-based company that wrote a program to let professional bingo players play dozens of bingo cards at the same time. The program runs on an IBM PC. Blanchard was ready to start marketing his system, along with custom-built tables and personal computers, to the big-time bingo halls, when one of his competitors led suit against American Multi-Systems for patent infringement.*
>
> *The case should have been thrown out of court for two reasons, says Blanchard. For starters, he says, his competitor's patent covered a hand-held calculator type device, not a general-purpose computer running a program. And the patent couldn't cover general-purpose computers, he says, because programs that play bingo are commonly written by students in introductory computer science courses. There was simply nothing novel or new about the technique that the patent described, and novelty is a basic requirement of patentability.*
>
> *Nevertheless, says Blanchard, the company that held the patent was able to convince a judge to grant a preliminary injunction that took American Multi-Systems's product off the market.*
>
> *Eventually Blanchard discovered a critical piece of "prior art" – concrete evidence that the invention described by the patent had been thought of before by somebody else – and was able to convince the judge to lift the preliminary injunction. Unfortunately, by that time American Multi-Systems had effectively put itself out of business with legal fees.*
>
> *"Judges are not particularly literate in technical issues," says Blanchard. "When they see a patent they presume that it's valid, as they should. (To them), if the Patent Office says that this is a valid patent, well, of course it's a valid patent."*

Patent lawsuits are a mechanism used by big companies against small ones[6]:

> *Among the executives and attorneys that we we talked to, the only candid description of how any big company uses the legal process to its advantage was offered by Mr. Johnson. On the subject of Lucent Technologies, whose aggressive "East Coast" style is becoming notorious in the Valley, he says, "They're incredibly aggressive in enforcing their Bell Labs patent portfolio as a profit center. They're regarded in the Valley as extortionist," he says. "Lucent's basic approach is to say, 'We're Bell Labs, and we have hundreds of thousands of patents, and we're sure that one of them must affect you and you must have infringed upon it. We'll give you a blanket license for all our patents for a limited period of time for a large amount of money, and if you don't pay, we'll sue you until the cows come home and you'll never see your children again.' There's no interest in any kind of strategic relationship. They're there to shake you down for as much cash as they can get." Lucent officials declined to comment.*
>
> *Red Herring, May 1999, written by Luc Hatlestad*

9.1 Defensive patents

As the patent systems allows for very general patents, a number of companies use patents on simple techniques to make it difficult for others to enter in the same market. Other companies, to defend themselves in case of being sued, do the same thing: they obtain patents on simple techniques. That way, in case of being sued, it's very probable that the plaintiff is infringing its patents and counter-sue him. As trials are very expensive, both sides get benefits **cross-licensing** their patents. The patents used that way are called **defensive patents**. This is very well explained by Oracle in the document sent to the Patent Office when it requested opinions on sofware patentability in January 1994[23]:

> ***Oracle Corporation***
>
> *Patent Policy Official policy statement issued by Oracle Corporation*
>
> *Oracle Corporation opposes the patentability of software. The Company believes that existing copyright law and available trade secret protections, as opposed to patent law, are better suited to protecting computer software developments.*
>
> *[...]*
>
> *Unfortunately, as a defensive strategy, Oracle has been forced to protect itself by selectively applying for patents which will present the best opportunities for cross-licensing between Oracle and other companies who may allege patent infringement.*
>
> *[...]*
>
> *Oracle filed its first patent application in November 1991, not because it felt that its software was suddenly worthy of patent protection; it filed that application because of concerns that other inventors, afforded patent protection by a flawed patent system, might find themselves in a position to seriously weaken the Company s competitive edge by alleging patent infringement. Even if Oracle had developed a certain invention first and could produce the appropriate prior art to prove its case, thousands of dollars in attorneys fees and other expenses would be spent in defense of its rightfully-owned technology. Oracle consequently believes that it must have a patent portfolio with which to respond to potential aggressors, so as to settle with them by cross-licensing to avoid litigation. Oracle is forced to channel a significant portion of its financial resources into patent protection of its assets, rather than using those resources in further innovating and expanding its computer software products.*

10 Conclusions

Software patents are, in general, harmful to the software industry and, in particular, to the European one, which is currently very weak, being composed of small and medium companies. These are what we feel to be the key reasons:

- Software patents tend to reduce competition, instead of promoting it, thus favouring big companies (non-European) that own a large number of patents and have specialized legal teams.
- Unless measures against obvious patents are taken, the European Union will be obliged to accept a multitude of trivial patents, requested by the big non-European companies that will turn into a permanent menace for the European small and medium companies, due the the high probability of accidental infringement and for the huge difficulty of programming while searching for patents for every problem which is to be solved.

[23] Full text can be found on the Internet at http://lpf.ai.mit.edu/Patents/testimony/statements/oracle.statement.html.

- The production of compatible programs within the EU, by European software companies, could effectively be blocked by non-EU companies, thus preventing any possibility of inter-operation.
- The developement of applications with similar functionality to others would also be hindered, making difficult or even preventing competition. This is specially grave when talking about electronic commerce in Europe.
- The *success of a software company is based in the **usefulness and quality** of its programs, not in its **brilliant** ideas*. So, the European Union efforts in order to protect its software industry should be directed towards the education of good professionals and companies, promoting open technologies and obstructing the adoption of those that lead to foreign dependence.

11 Acknowledgements

We would like to express our gratitude to Joel Spolsky, for generously having allowed us to include an essay of his in this report; to Roberto Lumbreras for his interesting ideas and to Eduardo Artuña, Emilio Albesa, Ángel Álvarez, Gregorio Fernández, Guillermo Díez, Jose Manuel García and Susana Moreno, that have reviewed this document and contributed to make it clearer.