

# Software is Patentable by Nature

*Observatoire Economique du Droit (UVSQ)*

**Philippe Simonnot, Director**

# Software is Patentable by Nature

White Paper by the *Observatoire Economique du Droit (UVSQ)*  
directed by Philippe Simonnot

---

The Council of the European Union adopted the following proposal on 7 March 2005:

*Member States will be obliged to ensure that their national laws consider inventions implemented by a computer as belonging to a domain of technology. To be patentable, an invention implemented by a computer must be new, susceptible of industrial application, and involve an inventive activity (which means making a contribution to the state of the art that does not exclusively involve a non-patentable subject).*

*According to the European Patent Agreement, a software programme as such cannot constitute a patentable invention. Inventions consisting of software programmes, whether expressed in source code, object code or in any other form, and implementing a business method, mathematical method or other method and not producing technical effects beyond the normal physical interaction between software and computer, network or other programmable device on which they are run, are not patentable.*

*Nevertheless, the Council introduced a new provision to specify that in certain circumstances and under very strict conditions, a patent may be associated with a claim for a software programme, alone or on a physical medium (e.g. CD). The Council considers that this provision would reflect current practice both in the EPO and in the Member States.*

For many years the question of the patentability of software has occupied minds and engendered passionate discussions, as is always the case when technology opens new economic opportunities. To contribute to the European Parliament's second-reading discussion of the proposed directive on computer-implemented inventions, as endorsed by the Council on March 7, 2005, the *Observatoire Economique du Droit (Université de Versailles Saint Quentin)* has published this White Paper which, since the *Observatoire* has no ideological view on the issue, simply attempts to lay out what economic logic dictates regarding this difficult subject.

The White Paper's conclusions are as follows:

- 1) The line of argument followed by various parties in the discussion is, usually, based on consequences. It tries to forecast the consequences of patentability or non-patentability of software on innovations -- their rhythm, their character -- and, therefore, also on economic growth, not only at national level, but also globally, considering the effects on developing countries. In our view, no objective conclusion for or against patentability of software can be drawn from such an approach.
  - 2) As a matter of fact, the present debate regarding software patentability finds an echo in the classical terms of the debate on patent itself, since it was created at the end of the 18<sup>th</sup> century. Patents were immediately recognized as constituting a fragile compromise between the necessity to encourage innovation and the fear of creating undue monopolies. Classical economic theory confirms the limitations of this approach to patents.
  - 3) The basic European text on software patentability (the Munich Convention) reflects this dilemma, which is further confused by a tendency to assimilate software to the domain of ideas. This has led to the well-known formula that computer programming "as such" is not patentable. It is this approach that was endorsed by the Council on 7 March 2005.
  - 4) Practice has departed significantly from the letter of the law: tens of thousands of software programmes have been patented in Europe and world-wide. It is thus evident that strictly speaking, the law has not been well-suited to the particular nature of software.
  - 5) By nature, software is patentable as a logical consequence of the economic theory of property. The Council was right to add a new provision to the existing legal framework.
- 1) The line of argument followed by various parties in the discussion is, usually, based on consequences. It tries to forecast the consequences of patentability or non-patentability of software on innovations -- their rhythm, their character -- and, therefore, also on economic growth, not only at national level, but also globally, considering the effects on developing countries. In our view, no objective conclusion for or against patentability of software can be drawn from such an approach.**

#### **1.a. *Axiological Neutrality.***

Before going into an analysis of the consequences, it is important to specify that the *Observatoire Economique du Droit* attempts to be axiologically neutral. It does not intend

to take any position regarding the aims pursued by governments, companies or individuals. Instead, our argument is focused mostly on the effects of patents on innovation. We must state that for us, innovation does not constitute a positive or negative value. There is, of course, a large body of literature in which invention has a pejorative connotation. In the twelfth century, for example, the term "invention" referred to a discovery, miracle, trick or lie. Joan of Arc, one must remember, was described as "a liar, troublemaker and inventor of miracles and appearances".

Many economists studying the problem of patents and software also claim to be neutral values. However, values turn out to come back to the discussion unnoticed. For example, Jean Tirole, an important contemporary French economist, wrote in a report to the Economic Analysis council of the French Prime Minister regarding intellectual property<sup>1</sup>:

"The report will avoid using vague notions of ethics and morality and will focus only on more concrete notions, such as efficiency and distribution... Specifically, the reason why abstract formulas such as those used in the theory of relativity must be excluded from the field of patentability has nothing to do with the fact that they are formulas regarding "laws of nature", but is related to the consequences such patents would have on users of laws (and to the fact that basic research is, usually, financed from public sources). Patentability ... must be analyzed in terms of efficiency ... and impact on re-distribution, and must not be determined by any taboo regarding the field of patentability. Any protection of intellectual property implies the privatization of knowledge that is part of mankind's common heritage. It is, therefore, fundamentally pragmatic and must be based on analysis of social costs and benefits".

While efficiency and distribution are, perhaps, more "concrete" notions than ethics and morality, they are not more precise: What is the measure of this efficiency? At whose expense should (re)distribution be made? Moreover, a taboo appears very soon, since on the same page, Jean Tirole writes: "Certain innovations (for example, those regarding drug manufacturing or certain forms of genetic manipulations) can be considered harmful at a given time by society". At a given time...! This is complete relativism. Why, for example, should weapon-related innovations, which are as dangerous as those involving drugs or genes, be patentable?

---

<sup>1</sup> *Propriété intellectuelle*, Reports of Jean Tirole, Claude Henry, Michel Trommetter and Laurence Tubiana, Bernard Caillaux, Comments: Daniel Cohen, Lionel Fontagné.

The doctrine outlined here by Jean Tirole, who represents a whole generation of economists, is explicitly “consequentialist”. This means that it judges patentability according to consequences in terms of efficiency and re-distribution, both on the national and on the global level. Tirole himself has said: “One can start from the reasoning that private and uncontrolled property of knowledge is not desirable either because it is inefficient (since it prevents distribution of knowledge or blocks the process of future innovations), or because it has harmful re-distribution consequences (against the poorest or against developing countries)”. If these words have any meaning, the last lines would suggest that “private and uncontrolled” property (what is private property worth if it is controlled?) can be taken away from its owner if it is estimated (by what Court?) that by its mere existence, it harms innovation and re-distribution. This is not a theoretical point of view. Obligatory licenses are, sometimes, imposed on patent holders, despite the right to enjoy private property (this provision has existed in France since 1968 but has never been used).

From the outset, we can, therefore, state that the position adopted by the prominent French economist is in fact ethical and moral with, in addition, many implied taboos. It is also strategic. Anyone who would oppose it would immediately be considered as an enemy of mankind, oppressing the poorest or impoverished nations, the term sometimes used to describe developing countries. We have dwelled upon Jean Tirole’s text because it is representative of an attitude that dominating in this field, which can easily be found in texts of various politicians addressing this issue.

### ***1.b. Compensating Innovation.***

In this regard, the problem is relatively simple: a patent’s only justification is to compensate innovation. However, from the moment when compensation is granted by society, things become complicated. Will “society” have its word in defining what inventions should be encouraged and to what extent? We must, therefore, define what is a patentable invention (as we know, according to the TRIPs description, it must “be new, involve inventive activity and be susceptible of industrial application”) and how much should be paid for it. Patent policy must strike the right balance between insufficient distribution of innovation because of monopoly powers granted to the inventor, and insufficient innovation caused by a lack of economic incentives to invent.

We must remember that software is a set of interacting programmes. Each programme is a sequence of instructions written in a well-defined language, encoding operations, processes or algorithms applied to data in the broadest sense. The programme is initially written in a programming language understandable to professionals. At this stage, it is in the form of “source code”, which permits the development and production of a software product. Once written, this sequence of instructions can be compiled, which means translated and transformed into a binary sequence (of 1s and 0s) signaling the

presence or absence of an electric impulse, comprehensible for a machine, but no longer to a human mind. It is then in the form of an "object code" or binary code.

Software invention is part of the general patentability problem: encouraging the inventor. Neither excessively, nor too little: it is a question of fine tuning.

There are two approaches to the innovation criterion. Proponents of software patenting are lined up against the militants of the Open Source software cause, who support free communication and use of source code. Proprietary software owners do not disclose their source code. For open source software advocates, software patent is not indispensable for innovation, and can even be a handicap. For proponents of proprietary software, the position is completely opposed: there can be no software innovation without patents.

The positions are often drastic. For example, General Mining Council, a very respected institution in France, does not hesitate to claim that software patents would have negative impact on innovation and competition. The *Comité de coordination des sciences et des techniques de l'information et de la communication (CCSTIC)* – a department of the CNRS – takes the same view<sup>2</sup>.

### 1.c. *Is Copyright Sufficient?*

For open source software advocates, patents are not indispensable, since the protection provided by the copyright is considered sufficient. By the way, neither of the two sides questions copyright and we can explain why. The author of open source software does not give up his author's rights to the software he wrote. We must note that exclusive rights granted to the holder of author's rights are much broader than those granted to a patent holder. The rights-holder may oppose any exploitation of his software, including exploitation for research purposes, while the patent holder cannot oppose its exploitation for commercial purposes. Patent protection is much shorter (20 years) and limited to countries in which the patent is recognised, while the author's rights are universal and valid for 70 years. However, we must stress once again that copyright protects the text, not the invention – the expression, but not the functionality -- because functions can be coded in a variety of ways.

Undoubtedly, copying of binary code is very easy and its cost is minimal. The imitator does not need to repeat the same stages of programme organization, writing and compilation, or to have access to the source code or description of all operations performed by the software. However, copyright is meant to discourage copiers. One can assume in advance a certain number of illegal copies, in the same way as a supermarket

---

<sup>2</sup> *Le Monde* of October 17, 2001

calculating a certain percentage of shoplifting in its operating costs. One can even consider that numerous programs would not sell as well if they could not be copied! Technical solutions exist for preventing overly easy copying. If they are not used, it is because, to some extent, protected software is less valuable than software that can be freely copied, because of the "network effects" the former generates. We will discuss these effects further. Moreover, when the price is low enough, it may be better to buy than copy.

Software patent advocates do not share this view, since the inventor is always exposed to another risk besides that of having his work illegally copied, namely the risk that another programmer would develop an innovation close to the original without violating the copyright. What is stolen here is not the text, but the invention itself, rendered in a different writing or different code. If the original invention is not protected by the patent, competition that destroys profits and discourages inventors will result. The patent helps to define, more or less explicitly, the perimeter beyond which a competing innovation is considered to infringe the protection granted by the patent.

#### **1.d. *The volunteer system***

Patent opponents claim otherwise. Passionate computer programmers, researchers and university workers are not interested in compensation. Honor, peer recognition and career prospects are sufficient to motivate them. Patent opponents use the argument that the software industry is, essentially, a labor industry, which requires little capital. Moreover, it is estimated that sixty to eighty percent of total R&D costs of sophisticated software is spent on software debugging and gradual re-writing of the code. This development activity would be performed efficiently by networks of programmers providing critical and independent vision within the project. This approach constitutes the basis of the so-called "open source software community": software is distributed with its source code, which permits developers to find and correct software bugs during development. The career prospects of the best programmers would be favored since their capacities would be evaluated publicly thanks to disclosure of source code and work that has been done on it. The "open source software community" claims to be particularly efficient in the fine-tuning of sophisticated software.

Availability of source code permits it to be examined by multiple independent experts, with more efficient error detection and faster correction of these errors. The "open source software community" has a contagious aspect: anyone who redistributes the software, with or without modification, must also transmit the possibility of copying or transforming it, whilst preserving copyright.

Software patent proponents may ask, how can we believe that an activity that generates billions of dollars of sales annually can be based on volunteers? And anyway, if the

client does not pay patent royalties, he pays for services which may result in much higher Total Cost of Ownership, as it is described in the American literature. Nothing is free in this world and open source software may be free, but it does not cost nothing and must be distinguished from “freeware”. Open source software is protected by copyright, as we explained, and is subject to specific licensing rules (generally, GNU GPL). Companies producing it sell personalized training, support, maintenance, tailored to specific clients’ needs as part of the package.

The money circulating in the software industry must end up somewhere. Either the programmer works for a company and all the added value of his inventive work is collected by his employer, or he is self-employed and then the client appropriates whatever surplus remains. The client may be another programmer, a company, public sector or a foreign government. It is not certain that open source software militants would always be aware that they are working for an oppressor, which may, one day, turn out to be a dictatorship government. In theory, the open source software community could function indefinitely if all the citizens of the world were programmers and, at the same time, inventors. However, since this is not the case, there are leaks in the system and non-programmers or non-inventive programmers profit from the work of inventive programmers and, in long term, the system is not sustainable. If open source software advocates put so much pressure on governments to adopt legislation favorable for them, it is, perhaps, because they have a feeling that without governmental help at least on the legislative level, but also, if possible, through public procurement, then the community is condemned to languish<sup>3</sup>. In fact, the majority of open source software projects originate more or less directly from the cloning of proprietary software. As a result, open source software cannot become a dominant mode, with proprietary innovation disappearing entirely. In reality, there is a sort of asymmetry between these two opposite modes: by definition, the source code of open source software is more accessible than that of proprietary software, so detecting a violation of legal protection is easier.

### **1.e. *The Most Efficient Owner.***

For patent advocates, the patent has the additional merit of transferring the monopoly situation that it creates to a more efficient actor. This is one of the virtues of the laws of the marketplace, defined very well by Ronald Coase, Nobel laureate in economics, in his famous theorem: the final owner of a property right, regardless of the situation at the beginning, is the one who is best placed to exploit it.<sup>4</sup>

---

<sup>3</sup> DeLong James V., *The Enigma of Open Source Software*, The Progress Freedom Foundation, March 2004.

<sup>4</sup> Ronald Coase won the Nobel Prize in 1991 for only two articles, one, *La nature de la firme*, written in 1937, resulting in the concept of transaction costs, and the other, *Le problème du coût social*, published in 1960.

### ***1.f. The Question of Circumvention.***

Moreover, the mere existence of a patent permits us to act on the expectations of the economic actors involved. A potentially market entrant could be completely discouraged from developing an imitation or an innovation that is too close technically and would be invalidated by an existing patent. Therefore, the patent obliges candidates entering the market to develop true innovations – as long as it is possible to define what “true innovations” are. Patent opponents respond that these “circumvention” strategies can be needlessly expensive for society as a whole. In classical industries, the need to circumvent may constitute a fundamental source of progress, leading to new inventions. Philippe Aigrain, when he was Head of Sector for Software Technologies in DG Information Society at the European Commission, before heading Sopinspace, claimed that in the software industry, the circumvention effort was not leading anywhere. A lot of money was spent in hope of collecting royalties that never turned out to be sufficiently high. For him, the system of software patents was a losing game, where the same work was financed repetitively.

### ***1.g. The “Network” Effect***

Patent opponents use “network effects” to support their position. This barbaric name refers to a phenomenon that is not new, but has taken on huge dimensions in the software industry. The idea is that the value attributed by a consumer to an asset depends on the number of users of this asset. Just as an isolated railroad cannot function if it is not connected to the network, a computer or software that is not compatible with those used by the majority of other users, declines in value. Network effects are well known in the transportation and communications industries, where competition between companies consists of extending of their networks and where the value of any given network increases considerably if it can be connected to other networks. However, in these fields, the network can be extended over the whole world. The “network effect” is nothing else than economies of scale applied to demand.

In the classical heavy industries, economies of scale related to production are sooner or later counter-balanced by “dis-economies of scale” – or, in other words, negative economies of scale caused by the difficulties associated with managing enormous organizations. This explains the fact that none of the automobile giants has ever conquered the entire world market. Thanks to the progress of software management, these limits can be extended, but not eliminated. In the information economy, the economies of scale related to production are relayed by economies of scale related to demand (network effect), which never stop growing because the market becomes larger.

If everyone uses Microsoft Word, it is yet another reason to use it for newcomers on the market.

The network effect has even been quantified in "Metcalfe's Law", named after its inventor: network value increases at the rate of the square of the number of users. If, for example, the value of a network for each user is equal to 1E, and there are one million users, the total value of the network is one million E. If there are ten million users, the total value of the network is around 100 million E, and so on. This law, which obviously, has no scientific basis, even though it can be observed for certain segments of certain computer industry markets, gives a convincing image of the network effect.

The "network effect" would protect investors better than complicated and extremely costly legal procedures. The "network effect" by itself will result in concentration on several software programs. Therefore, a given software can monopolize a certain segment of the market exclusively because it was introduced first, and the monopolization can be very fast. If the value of standard software depends on the number of users, then compensating an object whose value depends only on the number of users and not on its intrinsic merits, by means of a patent, does not make economic sense. This would be equivalent to claiming that technological development is a matter of chance.

Competition in this type of a market assumes an extreme form called "tipping" in the American literature (the "tipping line" resembles a water divide line: one can be on one side or on the other). Software that enters the market first is priced very low in order to prevent the appearance of competing products on the market, and thanks to the "network effect", it wins the lion's share of the market. Monopoly benefits are assured this way. In such a context, competing software tries to enter the market as open source software: thus Linux facing MS Windows, and Netscape facing MS Explorer tried to impose a new standard. In this case, the open source software appears to be an element of commercial strategy analogous to dumping. This strengthens the position of the software patent advocates, who claim that there is no monopoly in nature, that any existing monopoly is based one way or the other on the State (which is the case for a patent), that so-called dominating positions are fragile and can be reversed overnight if they are not protected by patents. Such upheaval can have a high "social cost", since the victory of the new software would not be due to its intrinsic qualities, but instead to the fact that it had made better use of the network effect than its competitor in sensing what distributors would be looking for, trends, etc. Once again, we get onto shaky ground where a "justice of the peace" (who, and according to which criteria?) would be asked to determine in advance which innovations are "socially expensive" and which are "socially beneficial".

### ***1.h. Duration of Patent and Technique***

As for the duration of a patent (20 years), there is a marked disconnect with technical progress, which limits the effective lifespan of software to a few years only. The protection provided by a patent can thus be wholly illusory, which is another argument against the software patent. Its advocates would respond: if this protection is so illusory, why formalize it? As a matter of fact, the average patent lifespan does not exceed seven to eight years. Beyond this, few patent owners continue paying the royalties to the State that are necessary to keep the patent effective (in France, € 300 per year and per country covered), simply because it is not worth it.

### ***1.i. Scope and Closing.***

Open source software advocates stress the fact that any software innovation is only a link in a chain and that a programmer inventing new software enters in good faith -- and without realizing it -- the area covered by many existing innovations, like a walker venturing out on the poorly marked land of his neighbor. This argument is two-fold, since it can be used to justify a more precise definition of property rights and their application to close the perimeter of the patented invention. At this point, open source software advocates could remark that if a programmer must negotiate an agreement every time he walks on the "neighbor's territory", it would, theoretically, lead very quickly to exorbitant transaction costs. However, these transaction costs must be borne by the client. In other words, the increase in transaction costs is limited by the market forces.

### ***1.j. The problem of Small Businesses.***

Moreover, from the point of view of the competition, protection granted by software patents is not necessarily more monopolistic than that of a company which acquired its position using only the normal commercial weapons. This is because the start-up, if it has an invention, has easier access to patent protection than to "commercial protection". The claim that patent protects positions acquired by large companies is often contrary to the reality. Without patents, the young aspiring companies which, by definition, do not have any commercial network, would not be able to survive or enjoy their eventual innovations. As Jean Tirole stated in the report quoted above, "[the] patent is not necessarily disadvantageous to small businesses"<sup>5</sup>. In reality, the opposite may be true. A small business, whether a start-up or not, needs to carve a niche for itself among the competitors. If it tries to sell its software to a large company, its negotiating power will, certainly, be higher if it has a proper patent.

---

<sup>5</sup> Jean Tirole, op. cit. p. 21.

The literature distinguishes two types of patents. The patent in its defensive form is meant to protect a new product, while an offensive patent gives its holder control over a new technical sector. Banks often request that a product on which a small business is based be protected by a patent. Filing, maintaining and defense of patents are, certainly, expensive, but these costs are often exaggerated. Preparing and filing a patent application costs approximately € 6,000 to € 10,000, plus € 300 for procedural costs. After one year, the candidate is offered a "PCT" (Patent Cooperation Treaty), which is international filing for world coverage, which costs 5,000 euros. Then, he may opt for protection of the European Patent Office, which covers 30 countries and costs 100,000 euros. Very few applicants go that far.

In the dialogue with lending institutions, patents permit small businesses to give more credibility to their projects. Fund providers do not, necessarily, have the knowledge to evaluate the technical merit of the case presented to them. Therefore, a patent validates a small business project, at least from the technical point of view. It has been repeated so many times, especially in France, that software protection was based exclusively on copyright, that many small business owners are discovering now that software patents exist and that they did not know about this in time to protect their innovations through patent filings. As we explained above, it is practically impossible not to penetrate the borders of existing patents. This risk is not specific to the software domain. There is a "balance of terror" between large groups, which, as the strategists know well, is a balance based on reason and often on out-of-court agreements and transactions. A small business without any patents cannot become part of this game of deterrence. In other words, instead of distorting competition and concentrating all cards in the same hands, software patents allow a leveling of the playing field. To follow the metaphor of nuclear strategy, we can say that patents allow a deterrence of the strong by the weak. Just as a nuclear balance was eventually achieved, a balance of patents can occur. By the way, many small businesses have understood this since out of 16,000 patents filed in France, 13,000 are of French origin, 5,000 are held by 100 large companies and 5,000 by small businesses, according to the numbers provided by Philippe Cadre (Institut National de la Propriété Industrielle) at the discussion on software protection on March 19, 2002.

Therefore, contrary to what is often claimed publicly, software patents are not the exclusive prerogative of large companies. Moreover, it is less widely known that large software companies use open source software as a commercial weapon in this way.

This does not prevent open source software advocates from using the small business argument. When we develop a new application, they claim, 1% comes from us and 99% from the common pool of our predecessors. If the common pool stops being common, and can be appropriated through patents, innovation is no longer possible except for major participants capable of obtaining cross-license agreements. This argument

disregards the fact that the “common pool” of the past will remain common since the past cannot be patented.

### **1.k. *The Paradox of Durable Software***

Software is a durable asset. We can even say that it is a perfect durable asset since it does not wear off even while in use. It only becomes obsolete as the result of technical progress. As the already quoted American economist Ronald Coase showed, durable asset monopoly makes its own competition. As a matter of fact, after providing software to the users most eager to buy it, the monopoly tends to lower the rates in order to reach less enthusiastic customers; initial buyers may, then, delay their purchase if they expect this rate reduction<sup>6</sup>. The freedom to adopt rates to reach an increasing large population turns against the monopoly and erodes its profits, since the monopolist becomes a prisoner of low-rate expectations of his buyers. No patent can prevent such a process. Software patent advocates respond – well, OK, if the patent does not really protect, let us patent in peace! In reality, Coase’s paradox does not apply to software as clearly as one could think, since, as the French economist Bernard Caillaux noted in his contribution to the aforementioned report to the French Prime Minister, “Software companies can make marginal improvements to their products, with larger reliability and improved user friendliness, which permits them to ship upgraded products regularly”. Programmed obsolescence strategies can also be adopted<sup>7</sup>.

### **1.l. *Impact on Concentration.***

The situation is complicated by the fact that the choice for software inventors is not only between patenting and not patenting. They can also resort to the trade secret approach. If patentability of software were impeded or prevented, then this would be the choice for the majority. Even with the proliferation of software patents, the trade secret approach is an obvious option for the reasons explained below. From the point of view of concentration in the software industry, it has been claimed that both the patent and the use of trade secrets lead to the above. Therefore, patentability would not add anything in this respect. On the contrary, as discussed above, one could expect a de-concentration due to use of patenting by SMEs. We must clarify that *Observatoire*

---

<sup>6</sup> See Coase R, (1972) « Durability and Monopoly », *Journal of Law and Economics*, number 15, p. 143-149, which is not a conjecture: Bulow J. (1982), « Durable-Goods Monopolists », *Journal of Political Economy*, number 90, p. 314-332, Fudenberg D., Levine D. and Tirole J. : »Infinite Horizon Models of Bargaining with One-Sided Incomplete Information « , in *Game Theoretic Models of Bargaining*, Roth (ed.), Cambridge University Press, Cambridge, p. 73-98 ; Gul F., Sonnenschein H. and Wilson R. (1986) : « Foundation of Dynamic Monopoly and the Coase Conjecture », *Journal of Economic Theory*, number 39, p. 55-190.

<sup>7</sup> Bernard Caillaux, op. cit. p. 144.

*Economique du Droit* does not attribute any *a priori* value to concentration or de-concentration (once again, there is no monopoly in nature, and even a single company producing a given product or service would be in situation of competition since this product or service would be in competition with all the rest).

### **1.m. *Impact on Disclosure of Knowledge.***

In contrast with trade secrets, open source and patents promote the disclosure of knowledge. If such disclosure is the desired objective (based on a certain system of values), it remains to be seen which of the two options is better. If patentability encourages those who resorted to trade secrets to file for patents, then the objective would be achieved since, in any case, they would not have resorted to open source. It is true that even in case of a patent, a large part of the innovation incorporated in the source code could remain inaccessible. Information that is the most useful is hidden in the source code and access to this code permits constant improvement of software. Even open source software advocates hesitate to publish the source code. If no complex program can be written without infringing patents, one would try to keep the source code secret. The argument works the other way - if source the code of an open source programme is not known, then the software is not really open source.

### **1.n. *Juridification.***

One of the most serious reproaches against software patents is the juridification [*i.e., the process whereby industrial relations become increasingly subject to legal rules*] of the field, something that the experience in the United States suggests should be avoided. It is true that businesses are being formed that can be considered as actual legal enterprises. Their function is to buy and manage patents with the aim of systematically looking for potential conflicts with owners of software, be it open source or patented. However, there is no reason why such companies would systematically favor large organizations. SMEs could also benefit from this legal approach, or adopt offensive strategies. As for the supposedly very high costs of this juridification, the same comment can be applied to them as to the transactional costs, of which they form part - they are regulated by the market.

### **1.o. *Public Interest.***

If we look now from the perspective of the public at large, we must note that it is served better by proprietary software than by open source software, which is too sophisticated and demands too much after-sale service, something which is always very costly.

Patentability could, therefore, be a means of redirecting R&D towards “more socially-desirable” innovations, if this expression has any meaning.

### **1.p. *Interest of Europe.***

It is sometimes claimed that software R&D moves towards the United States and away from Europe and that to counter this movement the European Union should keep pace with the United States regarding software patents. The following reasoning contradicts this last point of view: if software patents become widespread, development of open source software would slow down in Europe since legal insecurity would become too significant. On the other hand, if the European Union maintains its restrictive position, patents filed in the United States would not be valid in Europe, software innovation would be easier and faster and the Old World would have a new chance in the software war. We have to remember that the most important market is the United States and that European software inventions, one way or another, must be validated in that market.

### **1.q. *Conclusion.***

The conclusion of Bernard Caillaux, already quoted, is interesting, since it confirms that a consequentialist analysis [*i.e., an analysis based on consequences*] cannot lead to any results. “In the absence of an empirical analysis of the importance of all these effects (he wrote), it is difficult to adopt any position regarding the extension of software patentability. Major innovations occurred before patent protection was adopted or even considered for such inventions,... empirical studies are still insufficient,... the measurement of innovation is contestable.” He also adds: “To our knowledge, no econometric study so far has been able to provide an unambiguous answer to the question of patentability.”<sup>8</sup>

The consequentialist analysis leads, therefore, to an impasse.

**2) In fact, the present debate regarding software patentability retraces the classical terms of the debate on the very concept of the patent, since the patent was established at the end of the 18<sup>th</sup> century. The patent immediately was considered to be a fragile compromise between the necessity to encourage innovation and the fear of creating unwarranted monopolies. Classical economic theory confirms the consequentialist impasse of this conception of the patent.**

### **2.a. *Privilege.***

---

<sup>8</sup> Bernard Caillaux, Op. cit. 128.

Historically, the patent is a privilege granted, by the ruling powers, in the public interest. For economists who believe in the virtues of free exchange in all domains, this intervention of public authority in the name of general interest is disturbing.

Also, in response to Lincoln's assertion that "The Patent System added the fuel of interest to the fire of genius," Michel Chevalier wrote on this side of the Atlantic that the patent was "an outrage against freedom and industry." The solution proposed by the liberals was simple: Industrial success, not state protection, guarantees the validity of an invention. Another author from the same era, F. Malapert, wrote in terms reflecting the approach of the open source software community: "The study of the facts shows that there is no justification for the enormous privilege permitting one man to prevent others from working as they like... It must be known that any monopoly is a cause of demoralization and, at the same time, an obstacle to the progress of arts and industry, in a word the progress of civilization."

## ***2.b. Classical Economic Theory.***

If we refer now to the way in which classical economic theory approaches the patent dilemma, we can understand better the impasses of the software patent debate.

We will present this theory here in the simplest terms to select only what is useful for our discussion.

We assume that the software industry works at a constant cost. In a system of coordinates where quantities are shown on the horizontal axis and prices on the vertical axis, the supply curve will be horizontal since the costs are constant. This would be curve  $C_1$  (see graph).

It will meet the demand curve which, like all demand curves, drops on the right (quantities bought by consumers are inversely proportional to the price), at the point  $E_1$ , called the balance point. Classical economic theory indicates that at this point, the software consumer "surplus" is at its maximum (triangle I on the graph). This means that at this point, the well-being of the company in question is as high as possible. Given that the costs are constant, and that competition is supposed to operate freely, at this point  $E_1$  the profits of the software producer are nil (we assume that the entrepreneur's remuneration is included in the costs).

The same entrepreneur who invested in software R&D makes an invention that enables the reduction of the costs of services associated with the software. In the absence of any patent and, therefore, of any protection for this invention, the supply curve will drop from  $C_1$  to  $C_2$ . The supply and demand interplay, which is assumed not to be disturbed or falsified, will move the balance point which drops to  $E_2$ . The quantity sold of services

associated with the software will increase while their price will diminish. The consumer surplus will increase, as will society's well-being, but the entrepreneur's profit still remains at zero. In these conditions, the entrepreneur, who is not a philanthropist, will not make the R&D investments required for the invention since he expects that there will be no profit for him, and the innovation will not happen. We would remain at point  $E_1$ , which means at the lower level of well-being than the one that could be achieved if the innovation had happened.

To remedy this situation, the patent is introduced. It gives certain royalties to the inventor for a limited time. The result is the following: For the patent duration, the entrepreneur may keep the invention to himself (supposing he exploits it himself). The supply curve stays, therefore, the same at  $C_1$ , but since his costs drop to  $C_2$ , the entrepreneur has a profit per unit sold equal to the difference between these two costs. The total profit is obtained by multiplying the profit per unit by the quantity sold (rectangle II on the graph). If he transfers his profit to another company, the result is the same, since the cost of the production of services associated with the software in question will still be equal to  $C_1$  since the royalties claimed by the inventor are added to the cost  $C_2$  of this new enterprise.

For the whole patent duration, the consumer surplus remains at the level it reached before the invention. As soon as the invention enters into the public domain, the supply curve actually drops to  $C_2$ .

The consumer surplus increases by the same amount, with no profit for all the enterprises. On the graph, this surplus is equal to the sum of surfaces I, II and III. Consumers are now the only beneficiaries of the innovation. Social well-being has finally increased, which justifies *ex post* [*i.e., calculated retrospectively*] the creation of the patent.

The problem is that during the whole patent lifetime, there is a loss of well-being measured exactly by triangle III on the graph. Those familiar with classical economic theory will recognize that this triangle is equivalent to net well-being loss inherent to any state intervention made in the form of tax or monopoly. It is called net loss, because everyone loses.

Regarding patents, this loss of well-being is the result of the fact that for the whole duration of the protection, only patent holder royalties (rectangle II) are added to the consumer surplus which remains on the same level as before the invention (triangle I). The sum of these two items (if they can be added) is lower than the consumer surplus once the patent duration is over. The difference is the net loss (triangle III), discussed above.

The whole question is now whether this loss of well-being is justified by the surplus of well-being, which will follow. The evaluation is similar to the steps taken by an investor

comparing future profits with present sacrifices. The problem arises from the fact that the arbitration is made not by an individual, solely responsible for his acts and choices, who can compare the value for him of a future asset with that of a present asset, but by “society”, which cannot be reduced in any way to a single individual reasoning entity, excluding theories of social anthropomorphism.

This tendency is reflected, for example, in the oft-quoted letter of Jefferson to McPherson: “Inventions cannot be an object of property by nature. Society can grant exclusive rights to the resulting profits, to encourage people to pursue ideas that may be useful, but society does or does not grant such a right, depending on its good will and no one can claim or invoke it.”

Based on this, it becomes evident that “society” has something to say regarding innovations that merit patenting.

### ***2.c. Tautologies.***

Regarding the definitions of patentable inventions, they appear strange to non-specialists because they give an impression of tautology. The invention must be useful (“useful” in the United States, “having an industrial application” in France), must not be evident for a specialist (demonstrating inventive activity), and must contain an element of novelty.

In short, an invention must be innovative and a novelty must be new. Moreover, it must be “useful” but only according to the applicable norms. This is an obvious sign that we have not strayed far from classical economic theory. This is not the place to make a complete criticism of that theory<sup>9</sup>. The patent problem is only one aspect of its shortcomings.

Regarding software, the control of society leads us even further into tautology. A computer program is not patentable “as such”. It must, at least in Europe, have a technical effect.

We only left one tautology (that the invention is inventive) to step right into another one (that the technique is technical). This new impasse is very disturbing and to a large extent, feeds the present controversy in the European Parliament. And with cause: electronic flows induced in the computer’s circuits by a program constitute a technical effect, and under these conditions, any computer program as such is patentable. Hence the necessity to define the criteria more restrictively. This led to another expression, used by the European Patent Office: To be patentable, an invention must have technical

---

<sup>9</sup> Please refer to *Economie du Droit*, V. I and II (Belles Lettres), and to *L’Erreur économique* (Denoel).

effects “going beyond normal physical interactions between a program and computer, network or other programmable device on which it is executed”, formula repeated by the Council of the European Union. Patentability presupposes that the invention in question contains a new technical contribution.

No basic justification has been provided by the [European Patent] Office for this exclusion from patentability for an absence of technical effect.

No explanation is given that would permit us to understand why the existence or absence of technical character separates innovations into those that are patentable and those that are not. We are reduced to assuming that the main motivation of the European Patent Office is to avoid excessive investment in the domains where R&D is already encouraged by mechanisms other than patent-related royalties. However, the denomination “non-technical” combines very different R&D activities, some of which [“]present specificity that could justify patent protection,” as was perfectly expressed by Bernard Caillaux, quoted above.<sup>10</sup>

Michel Rocard, MEP, who will not be the last to lament the tautology of European definitions, proposes to describe technical domain as a “domain of industrial application that requires the use of controllable forces of nature to obtain predictable results in the physical world.” It is somewhat similar to the Japanese definition of patentability: “Any highly developed creation of technical ideas through the application of natural laws,” which is considered somewhat easygoing...

**3) The founding text of the European Patent Office with respect to software patentability, the Munich Convention, reflects this issue, combined with the confusion caused by integrating software into the domain of ideas. Thus, the well-known expression: computer programs as such cannot be patented. This formula was reiterated by the Council of the European Union on March 7, 2005.**

### **3.a. Article 52.**

In light of the foregoing analyses, it may be interesting to read again Article 52 of the Munich Convention, which is still used as a reference in discussions on software patentability:

- 1) *European patents shall be granted for any inventions which are susceptible of industrial application, which are new and which involve an inventive step.*
- 2) *The following in particular shall not be regarded as inventions within the meaning of paragraph 1:*

---

<sup>10</sup> Bernard Caillaux, op. cit. p. 159.

- a) *discoveries, scientific theories and mathematical methods;*
  - b) *aesthetic creations;*
  - c) *schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;*
  - d) *presentations of information;*
- 3) *The provisions of paragraph 2 shall exclude patentability of items listed in that provision only to the extent to which a European patent application or European patent relates to those items, considered as such.*
- 4) *Methods for treatment of the human or animal body by surgery or therapy and diagnostic methods practiced on the human or animal body shall not be regarded as inventions which are susceptible of industrial application within the meaning of paragraph 1. This provision shall not apply to products, in particular substances or compositions, which are used to implement any of those methods.*

### **3.b. Using economic theory of ownership as a guide.**

If we want to find a rationale in this Prévert-like inventory, which is not very poetic by the way, we should use the ownership theory arising out of the economic analysis of law rather than the classical economic theory as a guide. Let us summarize what it is all about beginning with the very interesting letter from Jefferson to McPherson on society's control over inventions referred to above. In this letter, we can read formulations vowed to celebrity, which are literally going to spoil the debate on patentability: "the person who receives an idea from me receives knowledge without weakening mine; similarly, the person who lights her candle with mine receives light without weakening hers." The so-called free-software community might as well gain inspiration from this.

In contemporary terms, economists would translate Jefferson's letter as follows: an idea is a good that is non-exclusive as well as non-competitive:

non-exclusive because it is impossible to prevent a user from using it even if that user does not contribute to financing the good;

non-competitive, because the "consumption" of an idea by an individual does not diminish the quantity that remains available to others.

For contemporary economists, these two qualities of non-exclusivity and non-competitiveness represent what they call a collective good or a public good, and many conclude that this type of good can only be financed by the collectivity, even though we can easily demonstrate that private collective goods do exist in reality.

And we finally get to the following formula, which has also become commonplace: “Knowledge is a worldwide public good.”<sup>11</sup> Or else, “Any protection of intellectual property, whatever it may be, implies the privatization of knowledge which is part of humanity’s common patrimony.”<sup>12</sup>

Non-competitiveness has never prevented the commercialization of a product. Listening to music in a concert hall does not prevent the simultaneous enjoyment of another spectator. And yet there are private concert halls where private orchestras with independent musicians play.

The key point is exclusivity. The economic theory of ownership leads to the replacement of this term by “capacity of appropriation.” A good is appropriable if the benefit derived from that appropriation exceeds the cost of such appropriation.

This is the same as to say that ownership is not self-evident. It arises when the cost/benefit balance described above is positive.

Example: I plant roses in my garden. This garden is mine as well as the roses. Am I therefore the owner of their perfume, their beauty? If by ownership I mean that capacity to exclude others from the consumption of my good, then I could not be the owner of that quality of my roses unless I erect a fence high enough to prevent passers-by from enjoying it. The “cost” of such a construction—and not only the cost of erecting the wall—is obviously prohibitive. Therefore I disseminate, and can do nothing but disseminate, in the public domain that quality of my roses. I am probably the legal owner. But I am not the economic owner, the possessor.

The first condition for any object to be patentable is to be capable of appropriation. How could I patent something that does not belong to me? The problem with the common term is that it takes the thinking away from the question of ownership rights, which must be restated at the core of the discussion on the patentability of software programs.

Ideas are therefore not patentable, because they cannot be appropriated, and they cannot be appropriated because the cost/benefit balance of their appropriation would be negative. To reach that conclusion, there is no need to refer to great notions such as the patrimony of humanity or the worldwide public good, which derive more from the *flatus voci*.

### ***3.c. The rationale of Article 52.***

With that in mind, let us read again Article 52 of the Munich Convention.

---

<sup>11</sup> Jean Tirole, *op. cit.*, p. 14

<sup>12</sup> *Id.* p. 15

The exclusions of paragraph 2, subparagraph a, cover “discoveries, scientific theories and mathematical methods”, which can be regarded as belonging to the world of ideas, thus incapable of appropriation, and consequently not patentable.

The exclusion of subparagraph b derives from another protection device, that of “models.”

The exclusions of subparagraph c, that is “schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers”, lie on the same principle as those of subparagraph a. They also belong to the world of ideas, incapable of appropriation and therefore not patentable. Computer programs are somewhat included in this list by extension. They fit in this list well, and therefore belong to the world of objects, which are not capable of appropriation and therefore not patentable.

Subparagraph d follows the same rationale: what is less capable of appropriation than a piece of information?

The Prévert-like inventory suddenly makes some sense.

Paragraph 3 is, however, more difficult to interpret. It qualifies the exclusions of paragraph 2. What is henceforth excluded from patentability is solely an item “considered as such.” It is obvious that the expression does not only apply to computer programs. And it would be interesting to check whether this expression has been used for other fields than software. It seems to have been the case in the presentation of information and in games.

Nevertheless, we can read in paragraph 3 a willingness not to grant patentability to an item, which considered as such would belong to the realm of ideas that cannot be appropriated.

Paragraph 4 is based on a rather different rationale, which is that of the respect for life as such, whether human or animal. This rationale is beyond our topic. It lies on the rationale of exception or urgency, which can have an economic justification. If patent considerations prevent saving someone’s life, the loss is irremediable.

### ***3.d. Refinement.***

One recognizes that the wording of Article 52 is rather awkward and that some refining is required. The sole little words “as such” are the source of legal confusion, uncertainty and insecurity. Back in December 1997, a round table organized by the Union of European Practitioners in Industrial Property ended with this conclusion:

- i) the exclusion of “programs for computers” in Art. 52 (2) European Patent Convention should be deleted or at least clarified and
- ii) great efforts should be made to inform private inventors as well as small and medium-sized firms in Europe that patents are available for computer software.

**4) The practice has departed from the texts: tens of thousands of computer programs have been patented in Europe, as in other parts of the world.**

#### *4.a. Proliferation of software patents.*

One can easily find proof that present legislation on software patentability is confused by the number of patents that already exist in Europe for software programs. This is to say nothing of the software patents that one witnesses in the United States and Japan where, it is true, the legislation is less restrictive. We have thus witnessed an expansion in the area of software patentability that is altogether legal..

#### *4.b. Evolution of jurisprudence.*

The position has been evolving in Europe since the Schlumberger decision of the Court of Appeals of Paris on 15 June 1981, which overturned the Mobil Oil decision (Court of Appeals of Paris, 22 May 1973). The decision of 1981 established that the object of the claim in question could not be reduced to a treatment of computer data; it also included concrete measures raised in drillings. A process could not be deprived of patentability for the sole reason that one or more of its stages involved using a computer. Such a solution would, in fact, lead to the exclusion from the realm of patentability important inventions that require the intervention of a computer program. Why should the hardware be patentable and not the software?

This decision led patent specialists working in the data processing sector to realize that the exception from patentability of computer programs provided for by the Munich Convention no longer made sense. This followed the shift which began in the 1970s from industrial equipment based on a cabled logic to that based on a programmed logic.

The European Patent Office stuck to the same course until the IBM decisions T935/97 and T1173/97, which accepted the program-product claims. From that time on, the EPO put an end to all ambiguity, for it became clear that in accepting a program-product claim, the program alone was *de facto* approved.

In the United States, the evolution of USPTO practices preceded, in a still more radical way, the European evolution. It is true that on the other side of the Atlantic, the context had changed radically after the adoption in 1980 of the Bay-Dole Act 5 (1980), which encouraged the obtaining of patents for inventions sponsored by the Government, whether made by agencies, universities or private companies. University researchers were encouraged to leave their ivory towers, patent their innovations and create businesses—with, in general, a small monetary interest for the university. All at once the number of patents granted per year increased threefold.

At the beginning of the 1990's, there was a debate in the United States similar to the one unfolding today in the European Union. Software was not patentable unless it was part of the hardware. However, following a series of legal decisions, software became patentable (cf. USPTO's "Examination Guidelines for Computer-Related Inventions," from 1995). Very quickly, software became independent of hardware. In 1962, for the first time, one single software program achieved 1 million dollars in sales. Today 150 billion dollars' worth of software is sold each year in the United States.

In Europe, with the softening of the position of the European Patent Office, some tens of thousands of software patents have been issued and patent protection for software is therefore a well-established fact within the European Union.

Recent decisions on attribution and in the resolution of conflicts seem therefore to have legalized the patentability of software programs, including the programs as such. The EPO has even granted patents for commercial methods. In 2000, the decision TY0931/95 rejected a method for managing pension funds; but at the same time, patents involving an interactive procedure for selection of choices from a menu (EP756731) or for a unique writing format for all the management tasks of a business (EP02099907) were granted. In summary, the software patent exists in Europe without great restrictions.<sup>13</sup>

In brief, the EPO's examining procedure has evolved to one of greater acceptability of software programs, ignoring its governmental guardianship and despite pressure from traditional industries that watch with anguish as part of their added-value chain passes into a realm that does not depend upon the modes of protection that they had controlled.

Is it possible to go back? But what is to be done with all the software patents already granted? One cannot imagine retroactive laws. The Commission's proposal for a directive has arisen out of a need to avoid this dilemma.

**5) Through the economic theory of property, software is, by its nature, patentable. Also, the Council of 7 March surely added a new interpretation of the founding texts.**

---

<sup>13</sup> Bernard Caillaux, op. cit. p. 155.

### *5.a. The specificity of the software program.*

A program, which is a function or a procedure, but not, once again, an abstract idea, has a very particular specificity: its “manufacturing secret” can easily be preserved. The inventor of a program can launch it without revealing the source code and unveiling only the binary code, which is, in principle, protected by author's rights—copyright. Even in a patent application, it has been seen, the source code can remain hidden.

The distinction between source code and binary code is a blessing for people who want to sell software because it allows them to keep their software secret as well as protecting their investment through intellectual property while distributing only the binary code.

Binary code can be understood by a machine but is difficult for a human being to understand. Certain techniques of “decompilation” enable some understanding of the source code from the object code, but the process is complex, very imperfect and costly. One would imagine that reverse engineering could be automated, but it cannot. Compilation makes the binary code opaque and keeps the source code completely secret, above all if it is not accompanied by any comments.

The problem of revealing to the public is special for programming. For other forms of intellectual property, revelation to the public is an inherent part of the process. People may object to the fact of a proprietary work, but at least they can see it and work with it. An energy professional, for example, can examine the patents that concern him. A law professor reads articles and at a considerable latitude in order to make a loyal use of the ideas that they contain without authorization and without payment. A composer can be inspired by a piece of music. Only the programmer is obliged to look at an opaque structure, outside of the process in which it was produced.

This difficulty extends not only to the person who tries to work with the program itself, but also to whomever tries to write programs that interact with it. Programming is not an exact science. But all programmers do not always have the same views as to what is “the same,” and faced with a situation in which someone is trying to write an application that will use his program, he may have a different view of the benefits of openness. Why should he help another programmer write a concurrent program, which, because of smaller development costs, might ruin the original? Finally, programmers are confronted with the classic “free rider” problem, that is to say that each one is better off if he guards his secret code whereas all the others give it to their public.

The fact that software programs are often compared to the world of ideas derives from the fact that the difference in nature between information and the means by which it is implemented is very slim, while the same is more obvious for physical, mechanical or chemical systems.

### *5.b. The Economic argument for copyright.*

The economic theory of property easily explains the existence of authorial rights. It would be extremely unlikely that from a chance mixture of letters would emerge a poem by Victor Hugo. Put another way, the poem in question is easily recognizable, therefore it can be appropriated without charge.

The same is true for the binary code of a computer program that is put on the market. It can be appropriated and therefore qualifies for being protected by a right of the author. This right confers a moral right and a patrimonial right on the author of a program, of which the text is considered to be an original work covered by copyright. This kind of protection is automatic. From an economic point of view, patrimonial rights imply a right of ownership on the form of the program, characterized by the power to forbid copying of the program. The author thus obtains a monopoly in the exploitation and distribution of the program, to the extent that the utilization of a program on a computer presupposes a copy of all or part of the program in the computer's memory.<sup>14</sup>

As has been said, authorial rights do not protect innovation. They must therefore eventually be protected by a patent.

### *5.c. Capable of appropriation, therefore patentable.*

Now, the fact that the source code is not revealed at the time the program is put on the market. This aspect of the program is, therefore, also susceptible to appropriation. Since it cannot be protected by authorial rights, it must be protected by some other means, which can be nothing other than by patent. Ergo, the program as such is patentable, whether or not it is written into the hardware. Patent protection does not awaken the code plan; it is at a superior, functional level.

We have seen the case of programs for scientific calculation for which new results of applied mathematics have been developed, and no one can see why this type of innovation should not be patented. For example, the LZW algorithm was patented by Unisys, and granted to 135 companies which make use of it under license.

The basic principle of the right of ownership is that the owner is free to do what he wants with his property if such property exists. This principle must be accepted by the free-software community.

---

<sup>14</sup> Under European law, the right of the author over the program is governed by directive 91/250/CEE.

No one can oblige the owner of a program to divulge his invention, even though he can release the program to the public without divulging it.

If he chooses to divulge it, no one can prevent him from trying to make a profit from it. It is not a question of pondering on the eventual consequences, positive or negative, of divulging or not divulging. It has been seen that this discussion was leading nowhere. And it has been believed as a principle that neither knowledge, nor innovation, nor economic beliefs were values in themselves. It is a question only of applying the right of ownership to the objects to which it can be applied, that is, to objects that can be appropriated.

*5.d. The reasons for the success of free software.*

Since the patentability of software is in the nature of things and the practice of the European Patent Office has evolved to obey this nature, one can ask oneself about the success of free software. The fact is that the open source movement has numerous supporters.<sup>15</sup>

- a) Programmers and administrators of data processing companies, because free software makes their job easier; programmers love to write code and they want to launch it;
- b) Believers in free software who think that software and intellectual creations should not be treated as property;
- c) The intellectuals who predict a new, collective mode of production;
- d) Those who are skeptical about intellectual property, who correctly perceive that there is a problem, but think, incorrectly, that free software can prevent it. These people have received the support of
- e) Software users who see in this a chance to get a free ride and lower prices;
- f) Software developers;
- g) Large data-processing companies who use Linux in multiple ways: as a competitive weapon for selling their machines, as a weapon against other producers of software;
- h) Distributors of open source programs, in order to sell their services;

---

<sup>15</sup> DeLong, op.cit.

- i) Ideological interests of 21<sup>st</sup> century socialists, seeking to ensure the collective ownership of key sectors of the economy;
- j) Anti-capitalist zealots;
- k) Those who hold anti-American views, who see the software industry as dominated by the United States.

*5.e. The Council complies with the nature of software.*

The Council of the European Union on 7 March knew what it needed to do to extricate itself from this discord. As has been said, it introduced a new provision in order to specify that, under certain circumstances and very strict conditions, a computer program can be patentable either as a standalone or as support. The Council believes that this provision would align the Directive with what is actually the current practice, in the EPO as well as the Member States.

The Council is complying with the nature of software.