

# Note on the patentability of software and of computer controlled inventions

(19/04/2005)

The purpose of this note is to define a consistent conceptual and legal framework in the context of the directive on the patentability of “computer implemented inventions” which can contribute to make the Community the champion of the knowledge economy according to the goals of Lisbon. It seeks to define a compromise that would provide innovators with adequate incentive mechanisms, while minimizing constraints to the freedom of entrepreneurship and to the diffusion of knowledge.

## **Purpose of the directive**

The directive on “computer implemented inventions” is a consequence data processing and information technologies entering all technological domains, which resulted in different practices among patent offices, some refusing the extension of patentability to software and intellectual methods, and others following the American practice on the matter.

A harmonised framework on this matter needs to be defined within the Community. The impassioned debate which took place during the first reading at the European Parliament made it possible to clarify a number of key questions, whose answers will shape the directive:

1. Should software be patented, as is done for instance in the United States?
2. If so, how should the patents be formulated?
3. If not, how should a limit to patentability be defined which would exclude software in an unquestionable way, and devise a clear limit with respect to the ever increasing presence of software within patentable devices such as washing machines, car braking systems, mobile phones, etc.?

This document attempts to provide answers to these questions, and consequently, to propose orientations to amend the directive in a coherent way, and according to the international obligations of the Community.

## **Key notions**

In order to understand the semantic subtleties of the different wordings of the directive, some key concepts are defined.

## **Software**

Software is the formulation of instructions in a programming language that will result, when interpreted by a computer, in the computation of a result which depends on the data provided to the program during its execution. Software is an immaterial creation, since the fact that the software be physically represented as holes in the surface of a CDrom, as magnetic fields on the surface of a hard disk, as electric charges in the memory of a computer, or as electromagnetic waves when the software is transmitted via satellite communication, changes nothing in its nature. The same is true for written works, which can be represented as ink spots on paper sheets or as numerical data in the memory of a computer.

The analogy with a book is very strong. In the same way that a book is the formulation in an original way of a set of ideas and narrative methods such as “first person narrative”, “balcony love scene”, “twins being mistaken for each other”, etc., software is the formulation in an original way of a set of ideas, of algorithms, such as “alphabetical sorting in a list”, “displaying a progress bar as the user waits”, etc. As in a book, the commercial success of a software programme will depend on the talent of its author, who will or will not produce a coherent and pleasant work. Like a written work, software is the specification of ideas, but in specific languages which are unambiguous unlike human languages. These program texts can be interpreted by the computer which will execute their instructions, but can also be read by another programmer who will understand what their authors meant. In that respect, it is important not to be able to monopolise the ideas that underlie software, so as not to infringe on freedom of speech. Could one imagine an author wanting to keep others from writing first person novels?

As these immaterial works are created by similar processes, it is not surprising that lawmakers assimilated software with other works of the mind, and extended the system of copyright to cover them by means of Directive 91/250 for the European Community.

## **Authors' Rights**

Rights granted allow the author of a work of the mind (song lyrics, music, books, and also software) to benefit from a temporary monopoly on the employment of his work. The copyright system is inexpensive, as its protection is automatic (any created work is assumed to be protected), even if the registration of the work in specialised offices is highly recommended in order to have proof of existence.

In the case of software, the copyright regime protects programs “whatever may be the mode or form of their expression”, according to Article 4 of the WIPO Copyright Treaty. While copyright protects software authors against piracy by servile copy, it does not prevent another author from creating a new work inspired by a pre-existing work. Since

its beginning, new ideas have been incorporated in the software industry by copying and improving existing software and by cumulating innovations. This copy does not penalize the original author because, unlike for the material world, the time required for a competitor to imitate software is barely shorter than the time required to develop the original software, since software development is more time consuming than simply having the idea itself. In this extremely reactive market, this means that initial innovators have the time to profit from their products before some competitor shows up, and they can continue incorporating further innovations and marketing them faster than their followers. Moreover, a competitor can gain market share only if his product is of better quality, brings additional innovations, or is cheaper than the original software, which is only possible if the price of the original software was too high, as we have seen above the software development costs are identical. An innovator can continue developing only if he maintains a dynamic state of perpetual innovation, without which he will leave room for new innovators. This is how the software industry has been one of the most innovative and reactive over the past decades.

### **Patents**

Patents allow an inventor to obtain a temporary monopoly on the employment of his invention. As for authors' rights, patents were not created to benefit their holders, but to benefit society as a whole. By granting the patentee a temporary monopoly on the exploitation of his invention, their purpose is to allow him to be paid back for the development costs of his invention which can be extremely high in the manufacturing industry. In exchange, the patentee renders public the specificities of his invention so as to guarantee his invention over time and to encourage new discoveries. It should however be noted that the patent system has a cost, and that a patent constitutes an obstacle to competition and to the freedom of entrepreneurship. This means that in any given industrial domain, if an overall benefit induced by patentability cannot be proved, for instance because a more adequate and less costly solution exists, a patent system should not be implemented in this domain.

Regarding software, the expected benefits seem very low with respect to the negative effects seen in areas where they have been used, and in particular in the United States. The first point to note is that the software industry was innovative long before software patents existed, so that there is no need for incentive by exclusion, with the risk of creating monopolies continually reinforced by the network effects that are inherent to the software economy. If copyright prevents servile copy of any individual software programme, patents can block market access to any competing software by monopolising the concepts implemented in software, thus preventing an innovator from outperforming a competitor who has become less innovative but has exclusive access to his market by

means of software patents. Consequently, all of the economic studies carried out in the United States since the introduction of software patents show not only no positive effect, but instead show negative effects in terms of poorly justified counterfeit software litigations, and in diverting a significant part of the companies' research and development budgets, estimated between 10 and 15 %, to legal expenses. Moreover, while the commercial lifespan of software is very short, on the order of a few semesters, the length of software patents cannot be less than 20 years because of TRIPS, equivalent to a monopoly on roughly ten generations of technology and therefore highly anti-competitive. This encourages companies to seek revenues from existing software at the expense of innovation and of reactivity. The patent system, devised for the slow development cycles of the manufacturing industry, does not seem well suited for an economy of immaterial goods. The majority of stakeholders estimate that the system of author's rights, used in conjunction with guaranteeing industrial secrets by forbidding reverse engineering (except in the precise case of interoperability), is better suited for the software domain.

Forbidding software patentability is therefore to be recommended, and the next question to be considered is how to handle the software present in patentable hardware devices.

### ***“Computer implemented” invention***

This term, coined very recently by patent offices, has not been adequately defined. It is misleading because a computer implements a program, which could lead one to think that a program may in itself constitute a patentable invention. Since the proponents of this directive claim its purpose was not to tend toward an American-style system, it would be more appropriate to speak of computer-“controlled” inventions, or of computer-“assisted” inventions, when speaking of inventions that combine hardware and software.

### ***Technicity***

The examination rules of the patent offices of the Community stipulate that, in order to be patentable, inventions must be new, inventive (or “non obvious”), susceptible of industrial application, and that an invention must constitute a “technical contribution”, evaluated in practice by its ability to bring a “technical solution”, by means of a “technical effect”, to a “technical problem”, that is, belonging to a “technical domain” (or “technological”, according to the TRIPS terminology).

The current juridical uncertainty comes from the fact that, no definition of the “technical” term having been given, the practice of the patent offices has evolved to continually extend the perimeter of patentability towards software.

The rationale for this extension is that, if for instance a new washing machine whose

technical effect is to wash clothes better and whose controls are carried out entirely mechanically can be patented, why not patent a machine which does the same washing, albeit using a computerized calculator, and why not patent the computer program as well?

This argument is not consistent, as it does not distinguish between the part which is physically involved in performing the innovative process, and the part that controls this process, which can be implemented entirely by means of hardware devices or by software. In the case of a computer controlled invention, the technical contribution is indeed performed by the material part of the invention, because it is this very part which physically carries out the patented process; whereas the software part of the inventive apparatus, designed to control it, cannot have technical features. For instance, in the case of the new washing machine which washes more efficiently thanks to a new computer-controlled washing process, it is the washing process itself, characterised by the order of adding detergents, water and so on, which produces the technical effect, the solution of the technical problem of washing. This process control could also be carried out other than by software, while the process would still be patentable. Reciprocally, the software, isolated and run in a simulation environment, would run the same as it would within the washing machine, albeit without doing any washing. The technical features that justify patentability are therefore not linked to the software, and this latter can never be considered as technical, or else plain software could be claimed as well. A doctrine conform to the text of the Convention of the European patent is thus established: in the case of a material invention that uses software, the invention as a whole can be claimed on the basis of technical contribution provided by its material part, whereas its software part, "as such" not technical, cannot be a part of the technical features of the patent. These doctrines are moreover fully compatible with TRIPS, as the software field as a whole is simply not considered a technical field (see below).

Moreover, the limit between material and immaterial processes (that is, software, but also educational and business methods) can be defined simply. If one defines an "information processing method" as any method which manipulates digitally represented information (that is, irrespective of the nature of its physical substratum, see above), whatever its nature or the origin of which it represents may be (data having or not a meaning in the physical world), it is sufficient to guarantee that information processing methods cannot be considered as technical to exclude software from the perimeter of patentability.

In terms of the perimeter of patentability, indeed, only slight changes occur, as the following examples show:

- a new washing machine, characterised by an innovative washing process, will be patentable, whether it is controlled by computer or by hardware, because it implements a purely physical innovative process;

- a new ABS braking system, characterised by a novel use of sensors (axle angular velocity sensors and accelerometers) and actuators (brakes), will be patentable, whether its embedded calculator is hardware or software. In fact, as innovations in this field are based on a new way to acquire information from the physical world and to control actuators which are also in the physical world, there is always matter for patentability. This is also the case for electronic injection (as better injection results in better progression of the flame front within the cylinders), as for other equipment. This interpretation is compatible with that of the European Patent Convention, which stipulates in its article 52.2 that software, as such, cannot be patentable, even when it is embedded within an inventive apparatus, which does not prevent the computer controlled invention to be claimed;
- an echo canceller for a voice-over-IP system cannot be patented if the innovation resides entirely in the software. The entire physical layer remains unchanged and development costs are therefore significantly cheaper than in manufacturing. Hence, there is less reason to monopolise the innovation. Here again, article 52.2 of the EPC is respected, because in this case there is no innovation at the software physical substratum level, and no invention can result from software alone.

## **Writing of the directive**

According to the above, the main points requiring amendment are the following:

### ***Computer “controlled” invention***

In order for it to be clear that software will never constitute a patentable invention, it should be specified that the “computer implemented invention” term, that is present for instance in the EPO literature, is to be understood in the sense of computer “controlled”, or computer “assisted” invention.

### ***Technicity***

In order for the underlying concepts that lead software development never to be monopolised, it should be specified that **information processing does not constitute a technological field in the sense of patent law**. Without this, it would remain possible, as it has already been the case for some patents granted by the EPO, that innovations in the field of educational methods, electronic voting, or the principle of a progress bar would be granted patents.

It will be also important to **define in a positive way the concept of technological domain**, so that intellectual methods such as methods for the exercise of economic

activities ("business methods") or educational methods cannot be monopolized. The absence of definitions would amount to leaving this to the chambers of appeal of the European Patent Office, a non Community organization which would then be in position to create law, assuming a prerogative of which it is not invested. Moreover, this would lead to a moving jurisprudential definition of technicality, quite incompatible with the objective of legal safety clearly affirmed by the initiators of the directive.

The delimitation between the physical and the immaterial worlds, which constitutes a limit which is legally certain, economically justifiable, and morally acceptable between the patentable and the non-patentable domains, will have to be expressed in an as clear and unambiguous way as possible. In the first reading, the European Parliament based on a wording present in the German law, defined a "**technological field**" as an **industrial application domain requiring the use of controllable forces of nature to obtain foreseeable results in the physical world**. The objective of this wording is to indicate that there can be invention only in the material world, by the use of physical interactions to obtain foreseeable results in the physical world, as opposed to numerical information processing methods. This wording can be reused, and eventually explained in hearings.

In order for the case of an innovative software programme running on a non innovative technical device never to result in the granting of a patent, it should be specified that **the inventiveness (non obviousness) of the technical contribution will be assessed only according to the technical features** of the patent claim, and not of the features (both technical and non technical) of the patent claim as a whole.

### ***Compatibility with TRIPS***

TRIPS does not impose that software be patentable. What article 27 of TRIPS states is that patents must be available to all inventions belonging to any field of technology. The rigorous interpretation of article 27 of TRIPS is therefore that either the software field is considered a technological field and thus any innovative software can be patented, whether it performs trajectory computations or is a word processor, or the software field is not considered a technological domain, and thus no software can be patented. What TRIPS imposes is therefore only an "all or nothing" approach.

Strong arguments against software patentability have already been put forward, but other articles of TRIPS also go in this direction. Whereas article 10.1 of TRIPS explicitly stipulates that software must be governed by the copyright regime, which is indeed the case within the Community thanks to directive 91/250, article 13 stipulates in turn that one cannot systematically prejudice the legitimate interests of the rightholders of

copyrighted works. However, considering software as patentable would create an exception regime that would indeed prejudice authors, since the author of original software could be prevented from diffusing it on the pretext that this software would infringe one or more software patents. In order to guarantee the non patentability of software, it should therefore be mentioned in the directive that **the computerized processing of numerical data cannot be regarded as a technological domain in the sense of TRIPS and of patent law.**

This will never prevent computer controlled inventions such as ABS breaking systems or washing machines from being patented, as has been explained above. What was legitimately patentable will remain so.

### ***Form of claims***

In order to prevent the patenting of software and obstacles to freedom of information, **software claims** (claims of software in themselves or on any medium) **will be forbidden** both in **negative** (rejection of software claims) and **positive** ways (only claims on physical products or physical processes will be allowed).

Care will also be taken to indicate that **the production, handling, processing, distribution and publication of information in any form can never constitute a patent infringement**, direct or indirect, even when technical devices are used for this purpose. The patent system, initially intended to encourage the diffusion of information cannot be turned against it.

### ***Interoperability***

In order to enforce the freedom of reverse-engineering for the purpose of interoperability, but also that collected information can effectively result in market availability of interoperable products, it should be stipulated that, when **the use of a patented technique is needed for the sole purpose of interoperability, this use must not be considered a patent infringement.**