

Software Protection

1.- General overview.- Software industry with Microsoft, IBM and other big companies, but also with many developers, many small structures and that in all the countries around the world occupies a very specific and important place. The size of the worldwide software industry in 2008 was US\$ 303.8 billion, and more that is to say an increase of 6.5% compared to 2007. The truth is that US companies dominate the software industry: 74 companies out of the 100 largest software companies are headquartered in the United States. The European Union follows with 13 companies. 8% of the largest software companies are Japanese. But this industry is present in all countries, without exclusion, and for instance in Egypt or in the Emirates.

So Software Protection is a universal problem that all the countries have to deal with.

But Software Protection is also an old new problem... I say old new problem because the question now dates from around thirty years, at least in the countries where computing knew its first developments. Nevertheless it's a topical question. And that's the case because, even if the TRIPS agreement says in its article 7: "Computer programs, whether in source or object code, shall be protected as literary works under the Berne Convention (1971)", the question was renewed with the reintroduction, for ten or fifteen years according with the countries, of the patent in the approach of "protectability".

In fact, the history of software protection is a chaotic story. And the question of software protection appears today as a question of "oscillation" between the two poles that on the one hand copyright is and on the other hand patent is, with what implies in terms of complexity. It's necessary to wonder about the copyrightability of the software and about its patentability but also about the coexistence of the two systems of protection.

Of course the patent way has more substance in the economically developed countries where the patent is commonly used. But, whatever the country is, it's not possible to choose to overlook that one, if we want to have a general overview of our question. Consequently, I shall not content to adopt the TRIPS approach.

My plan will be naturally:

- Copyright;
- Patent;
- And copyright... and patent.

I.- Software protection and copyright

2.- The place of the copyright protection.- The software protection by copyright seems the pathway (we would say in French the “King Way”), today firmly established by the TRIPS and consequently by all the legal national systems. At the time of the end of Soviet Union, the United States made it one of their key issues. It is very interesting to recall that this point was one of the last points negotiated between the United States and the moribund Soviet Union.

But, in fact, if we make the effort to look into the state of thoughts in the seventies or at the beginning of the eighties, i.e. when the question began to be considered, in this time the commentators spoke more about patents. But the software was proclaimed no patentable in several countries for different reasons: for instance, in France, to avoid a US grip on the software industry and in the United States because of a narrow understanding of patentability at the time. So the copyright way was imagined by creative lawyers as a “spare wheel”, an “escape route” in the aim to obtain a protection for software in spite of the rejection of the patent.

The prime mover of this “discovery” of the copyright was the will to obtain a protection for the software industry. The basic idea was to grasp the software as writing. Software is written, surely in a very specific language, but written. And so, at least in this approach, it must be analyzed as a writing work. Why not? Even if, to say the truth, it’s a bit surrealist. Software is put somewhere between Victor Hugo and Naguib Mahfouz!

3.- Copyright vs. author’s right.- However strange the option may seem the fact is today that this way of protection is adopted in all the countries.

But, as everyone knows, we have in our earth two systems for the protection of literary and artistic works which are based on two different philosophies: copyright in the strict meaning of the word and author’s right. The boundary between these two systems runs across for instance Europe or Arabic World. United Kingdom belongs to the copyright system and France and Germany to the author’s right. Lebanon or Tunisia belongs to this author’s right system and Jordan more to the copyright

system. But, as far as the author is an important character in the system significantly called “author’s right”, even in the Arabic countries of the copyright family, the influence of the author’s right philosophy is perceptible. That can be surely explained by the humanist tradition of the Arabic culture. In fact, there are few countries which are at the frontier of these two systems. The Netherlands are a good example of this position. Canada is a very specific case because a part of the country is dominated by Common Law and the Province of Quebec by Civil Law. So the international landscape is objectively complex.

But why put the stress on these differences? The answer is easy. The copyright approach is more an economic approach, with an interesting view of dissemination of the works. The author’s right approach is more a humanist approach, with the author in the heart of the legislation and so a strong protection of this one.

As far as the software economy partakes of an industrial economy (of course information industry, that means a revisited notion of industry, but industry), resorting to copyright is easier than to author’s right. Copyright doesn’t know Victor Hugo (or Hemingway) and so it’s easier to link software and writings when the writer is not an important character. That doesn’t mean that author’s right would not be appropriated to the software. But that means that the application “as is” of the usual rules needs a specific adaptation.

In fact, in all countries, there is a specific approach, more or less significant, of software protection by copyright or author’s right. I propose you to put the stress on three main questions.

4.- Subject matter.- The first question is the basic question of “protectability”: what in concrete terms can be protected?

The first part of the answer is easy: protection of software means protection of the software form. That is very classical and similar in the copyright and the author’s right countries. It’s the intellectual creation which is protected (and not the support). But also, according to a universal principle expressed again in the TRIPS, “Copyright protection shall extend to expressions and not to ideas, procedures, methods of operation or mathematical concepts as such”. Ideas are free.

The second part of the answer is a bit more difficult. A work is protected only if it is original. The requirement is the same in all the countries. But the understanding of this word is deeply different from a country to another, and, more, within a country, from an interpretation to another! In order to concretize this observation, it can be observed that in the copyright countries original is often

understood as a quasi-synonym of “originating” from a creator, which is not very far from the idea of novelty. But, in the author’s right countries, and in the first row France and Germany, the word original is traditionally understood very differently. The originality is the print, the mark of the author’s personality. And we find that also in many Arabic countries. But it’s very obvious that this approach is not well adapted to works as software. It’s possible to find the “print” of the personality of the author in a poem or a piece of music. But is it possible to say the same thing about software?

The German High Court, in 1985, kept a traditional view and, according to the commentators, the consequence was a non real protection of software in Germany. In consequence, in 1986, the French *Cour de cassation* choose to define originality in other words, at least for software: the Court said that the originality must be understood as the mark of the personal contribution of the author. From that time, it’s also the European approach. That means that, in the case of software, it’s not necessary to scrutinize this software in the purpose to discover the hypothetical whole “presence” of the author in the software. It’s sufficient to notice a contribution of the author, i.e., in my opinion, choices made by this one. That is much less demanding.

The conclusion we have to draw from these quick observations is that, in a practical point of view, the requirement of originality must be surely revisited if we want to really give a protection to software. In a more theoretical approach, the “software object” implies in my opinion to reconsider the notion. Personally, I proposed to understand originality as novelty “in the word of the forms”. But it’s a very peculiar opinion!

5.- Protection beneficiary.- The second relevant question: who may have the rights on the software?, can be presented almost in the same terms.

The national traditions are very different. The copyright system knows the notion of “work for hire” which has for effect to give the rights to the employer in the case of employee creations. On the opposite side, a law as French Law keeps the rights in all the cases (almost in all the cases) to the author and so to the employee even in the case where the creation was made for the purpose of his enterprise – what is, to say the truth, deeply unrealistic. Between these two extremes, many laws, and in particular in the Arabic world, present a median position: they recognize the rights to the author who created the work but, in the case of an employee creation, they concretely entitle the employer.

Having said that, it’s interesting to note that, when it’s matter of software, these divergences disappear. In Europe, the Directive on software of 1991 gave to the Member States a common direction.

Article 2 (“Authorship of computer programs”) says:

“1. The author of a computer program shall be the natural person or group of natural persons who has created the program or, where the legislation of the Member State permits, the legal person designated as the rightholder by that legislation. Where collective works are recognized by the legislation of a Member State, the person considered by the legislation of the Member State to have created the work shall be deemed to be its author.

2. In respect of a computer program created by a group of natural persons jointly, the exclusive rights shall be owned jointly.

3. Where a computer program is created by an employee in the execution of his duties or following the instructions given by his employer, the employer exclusively shall be entitled to exercise all economic rights in the program so created, unless otherwise provided by contract.”

So it’s clear that each Member State can follow its tradition: recognition of the work for hire, recognition of the collective work; or can adopt another approach. It does not matter. The important is to reach the specified aim: to entitle the employer to exercise the rights.

And actually a country as France adopted a specific provision for software. So article L. 113-9 of the French IP Code says today: “Unless otherwise provided by statutory provision or stipulation, the economic rights in the software and its documentation created by one or more employees in the execution of their duties or following the instructions given by their employer shall be the property of the employer and he exclusively shall be entitled to exercise them”.

The conclusion is obvious: in the case of software, the solutions about the ownership are convergent. That is a considerable simplification in the context of a law suit.

6.- Right’s content.- The determination of the content of the rights corresponds to the same imperative of convergence but the matter is more complex. It’s not enough to say that rights exist on given software. It’s necessary to determine the exact content of these rights: what can do the rightholder? The question is obvious. But the answers are not. They differ according to the different national laws, even if, on the basis of the old Bern Convention and the modern WIPO Treaty, the two main rights are the reproduction right and the communication right.

Only the software specificity leads the legislator to rewrite the content of the rights.

Obviously it's not possible to present a general overview of the different national systems. But as far as the exact "impact" of a right can be assessed considering the exceptions this right suffers, in the present scope of this paper it seems to me welcome to underline two specific points regarding these exceptions, which, one again, express at the same time the particularity of software and a (relative) convergence of the national laws.

The first point I want to bring to light is the disappearance of that we call in different laws the "private copy" exception. In other countries, it's matter of private use. In the copyright systems (in the strict meaning of the words) the way is not the same; it's matter of fair dealing in England or of fair use in the United States. But, in all the cases, the result of these different rules is the same: a private use is not prohibited or, more exactly, cannot be prohibited. The truth is that the digital environment is not favorable to this exception. The fact is that, in the proper case of software, the exception was generally dismissed. For instance, the European Directive substitutes the right to make a "back-up copy" for the private copy.

The second very peculiar point is the recognition of a "power of intervention" on the software. That requires explanation. The starting point is this one: the elected right is, as we saw, copyright or author's right according to the national system but the software object is an industrial object or, at least, a practical item. That's the reason for which the question was asked first in the United States and after in Europe whether it was possible to organize a reverse engineering in the case of software. Reverse engineering can be defined as the process of discovering the technological principles of a device, object or system through analysis of its structure, function and/or operation. It's clear that doesn't make sense in the case of a work as a novel or a movie. But the question can be effectively asked when the work is software. The answer is not very clear in the United States because some courts came to a positive decision and some others to a negative one. In Western Europe, a new and complex right was recognized, which received a new name: decompilation.

The best is to quote the European text, as it appears in the European directive, in its article 6:

"1. The authorization of the rightholder shall not be required where reproduction of the code and translation of its form within the meaning of Article 4 (a) and (b) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:

- (a) these acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorized to do so;
- (b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in subparagraph (a); and (c) these acts are confined to the parts of the original program which are necessary to achieve interoperability.
2. The provisions of paragraph 1 shall not permit the information obtained through its application:
- (a) to be used for goals other than to achieve the interoperability of the independently created computer program;
- (b) to be given to others, except when necessary for the interoperability of the independently created computer program; or (c) to be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.
3. In accordance with the provisions of the Berne Convention for the protection of Literary and Artistic Works, the provisions of this Article may not be interpreted in such a way as to allow its application to be used in a manner which unreasonably prejudices the right holder's legitimate interests or conflicts with a normal exploitation of the computer program.”

It's really difficult to consider this text as clear! That's putting it mildly!

But in our scope the main point we have to keep in mind is once again the specificity of the “software copyright”. It's a revisited copyright, and, if not a “cross-border copyright”, a copyright envisaged in the light of common challenges (common to the different countries).

Patent is also, in a sense, a common challenge...

II.- Software protection and patent

7.- From the patent rejection to the patentability.- As I said above, if the first thoughts were about patents, the “patent way” was closed very early. So it can be strange to present today the patent as a rival way for the copyright.

Nevertheless it's true. That suggests two questions: why to pursue such a protection? How to reach it? How to reach it if we consider the first state of non patentability?

8.- Why patenting? – Why this wish of protection? I think that two reasons can be identified. At least, *I feel* that there are two different reasons.

The first one is objective. It must not be forgotten that the copyright “grasps” the form: the writing, i.e. the programming “loops” and the general structure of software. But no more. It doesn’t give a right on the substance of the software, on the process itself. The firms wanted to have a more consistent control of the object software.

The second reason, still according to my view, is more based on subjective considerations. It’s important for a firm to have a portfolio of patents. But it doesn’t make sense to have a portfolio of copyrights! The effect of fame is not the same. The media effect is not the same. The patent has a proper appeal.

So, in spite of the first attitude of the legislators and judges of rejection of the patent, the firms turn towards the patent. In the eighties, it was clear that many applications at the European Patent Office were that we may call “test applications”. Their purpose was to try to see what the EPO was ready to accept. But it was very difficult to apply for a patent for software as such. It was necessary to use periphrasis. Things changed during the nineties.

9.- How Patenting? – Things changed... So the second pertinent question is: how this change happened? How it was possible to swing to a policy of protection?

If we widen the angle of view, we must observe that the question is not pertinent for all the countries and, for instance, for a country as Japan. Doubtless, Japan chose the protection by copyright. But, in the facts, the patent was never abandoned. And so it was not necessary to discover it again!

In the United States, the change was not specifically a change about the patentability of software but was a general change in the approach of patentability. Everyone knows the well known case *Diamond v. Chakrabarty* in which the Court said, in 1980, that can be patented « anything under the sun that is made by man ». The sentence is obviously excessive. But, excessive or not, it meant leaving the door wide open to a policy of patentability without real limits. It was only a beginning. Software was patented and later, as you know it, business methods too – what is an American specificity.

In Europe, evolution was a bit more complex because there was – there is – in the Munich Convention “on the Grant of European Patents” an article 52 which says that “European patents shall be granted for any inventions”, but, just after, in the second paragraph: “The following in particular shall not be

regarded as inventions: discoveries, scientific theories and mathematical methods” (etcetera) “and programs for computers”. So, in brief, by legal decision software is not an invention and that when the first requirement in order to obtain a patent is to present an invention. It seemed there was no doubt about the non patentability of software. But that was to ignore the definition progressively elaborated by the European Patent Office. For the EPO, considering the list of the “non inventions” of the article 52 (I quoted partially), an invention is a technical solution given to a technical problem. If software is ruled out of the patent protection, for the EPO it’s because software is a non technical creation or, at least, apprehended as non technical. It only remained for the Office to say that as far as given software produces a technical effect, it cannot come under the legal rule. In other words, software is not patentable because it is not technical... but if it is technical it can be patented. And there you have it! It’s expressly said by the EPO Guidelines: “While "programs for computers" are included among the items listed in Art. 52(2), if the claimed subject-matter has a technical character it is not excluded from patentability by the provisions of Art. 52(2) and (3)”.

10.- A patent as another? Once the software patentability allowed, it remains the question of the concrete issue of the patent. The point was discussed; what the good policy might be: to follow the common rules or to imagine specific rules? For instance, has the requirement of non obviousness (non obviousness of the invention) to be examine as usual or with a lower strictness as some authors suggested it?

The European approach is clear. It’s enough to read in the EPO Guidelines: “The basic patentability considerations in respect of claims for computer programs are in principle the same as for other subject-matter”. Why this restriction: “In principle”? Because especially the valuation of the technical effect – which is, as we saw, an important point in the view of the EPO – is envisaged in an original way. On the idea that software produces necessarily a technical effect when it runs on a computer, EPO requires a “further technical effect”. As we can read again in the guidelines, “a further technical effect which lends technical character to a computer program may be found e.g. in the control of an industrial process or in processing data which represent physical entities or in the internal functioning of the computer itself or its interfaces under the influence of the program and could, for example, affect the efficiency or security of a process, the management of computer resources required or the rate of data transfer in a communication link”. This notion of “further technical character” is unusual. But, in fact, that’s only an adaptation of the common model. Moreover it’s effectively the common rules which received application. The Guidelines say clearly: “If the subject-matter passes (the) prima facie test for technicality, the examiner should then proceed to the questions of novelty and inventive step”. That’s nothing else that the usual requirements for patentability.

So common rule? Common Law? Yes and not. Great offices as the USPTO or EPO accept to patent software as a product. But, if we consider that with a perfect freedom of thinking, it's very strange. Software is a set of instructions. In the patent categories, it's a kind of process but not a product. The explanation of this odd distortion is that a lawsuit in infringement is easier when the object of a patent is a product. But objectively that doesn't justify naming product a process!

But that puts the stress on what this kind of patents pretends to protect. The position adopted by the Offices doesn't clarify things. But it can be understood – and that in perfect coherence with the global philosophy of the patent system – that the protection is given to a technical solution (if I adopt here the vocabulary of the EPO). It's something else that the copyright subject matter.

That must lead us to bring together copyright and patent.

III.- Software protection between copyright and patent

11.- Concrete item and legal subject matters.- So we have, at least in some countries, two protections for a single item. Is it not... too much? Or, to be more serious, how managing the potential contradictions? The two legal systems, the two regulations are not the same. For instance, the exceptions to copyright and to patent right are not the same. To give a concrete example, in the different national systems of patent, by different ways, an “experiment exception” exists but copyright doesn't know this kind of exceptions which has no sense for a traditional work (a poem or a piece of music).

A very intellectual approach can be a way to solve or to try to solve this problem. We speak indeed, we spoke about software. But it's a simplification. A unique word covers two realities. If we would to be very precise, we would speak about the program writing and about the program solution. Program writing is the copyright object. Program solution is the patent object. In practice, in our field, the patent claims take the form of a method of operating said apparatus, the apparatus set up to execute the method, or the program itself. But in all the cases, it's clear that the legal objects are not the same if we consider the copyright system or the patent system. In this view, there is not a real accumulation of rights on a single object... because there are two objects.

But, as I said, it's a very intellectual view. In the real world (real world in contrast with... the legal world), this distinction has no sense. For a programmer, for a trader, for a practitioner, there is only one item!

And effectively, in a concrete approach, copyright and patent have to be managed in the same time.

It's not always easy.

For instance, if a potential counterfeiter pleads a legal exception (as the experiment exception I spoke previously), it's very important to untangle the problem: what this presumed counterfeiter exactly did? Is the act a matter for copyright or for patent? In a case of lawsuit, it is up to the lawyers to present to the Court a strong analysis of the situation. And it is up to the Court to build up its own analysis.

But an ordinary but also very efficient way for managing the coexistence of the two rights and for anticipating the difficulties is of course the contract. That means yet to conceive the contracts in consideration of the two rights: copyright and patent. That means to use the contractual tool as a mean for reducing the divergences – for instance for settling the question of the ownership or of the transmission of the rights in accordance with copyright *and* patent system. The drafters have to show they are imaginative. Interesting point: today, in the world of “free software”, the licenses are not only conceived as “copyleft” licenses. From now on they contain clauses about patent. For instance they restrict the faculty to claim a patent or to use a patent.

Complexity was increased.

And so?

12.- And so, to conclude, it's necessary to observe that this complexity is not equally shared. The patent issue is more, as I said above, an issue for the economically developed countries where the patent is commonly used. When it's not the case, things are simpler. Software Protection can be considered in the sole light of the copyright system.

But the two facets exist. And an international Center of arbitration as the Center of Sharm-el-Sheik (and others too...) must be ready to face the complexity. As I said the drafters of contracts must show their imagination, the arbitrators too.

Michel Vivant

Professor at “Sciences Po” Paris

Head of the Intellectual Property Speciality of the Master of Economic Law

[First year of PhD]

Doctor *honoris causa* of the University of Heidelberg