Component-based Software[i], Patents [ii]& Open source[iii]: a guideline through the Bermuda Triangle.

With commercial success come friends and enemies you didn't know you had.

But everything is relative. Success, failure, friends or enemies depend on who you are (industry, university, start-up) and on your business model.

Dealing with software, commercial success comes with a product that contains (embed) software or with software on a support (memory stick, CD-ROM) to be run on a device. For our purposes, and for reasons explained further, software and the device that runs the software will be assimilated, and considered as our product.

Having developed our product -or developing it- the question is never "can I get a patent or not?". The real questions are "can I get money out of it?", "how can I secure my business?", "what is its added value?"

Before having a more indepth look into these questions, why are we talking about all this?

In the early days of the computing industry, the value was in hardware. Software and source code came as part of the computer package. There were only a few programming languages in existence at that time.

As time has moved on, the value has resided more and more in software.

Today, value principally resides in software, services, and in data (which requires management by software), as any social network, SaaS (Software as a Service), PaaS (platform as a service) or any business intelligence professional knows.

Dealing with source code, as a matter of trends, it started with being accessible (but not yet open as we say today), then moved to proprietary, with a recent and increasing trend to open source. Nowadays, open source solutions are hugely developed, also by companies that used to deploy proprietary software. Patent holders deploy open source software and Open source organisations file patent applications, creating a complex mesh of interests, opportunities, and constraints.

Such complexity can be faced almost everywhere since software is found in smartphones of course, but also from professional concerns like medical imaging technology, navigation systems, car safety features (ABS, ESP), to domestic appliances such as Blu-ray technology, washing machines, refrigerators, vacuum cleaners, etc.[iv]Software are found in all market sectors, for all types of techniques, from high tech business-to-business (B2B) products to mass market products.

Our product can then belong to any one of these markets. It can be ready for release or still under development.

Developing software is a creative task. Yes it is. And creating software is an investment leading to value that requires, deserves protection. Appropriate protection relies on Intellectual Property (IP). Indeed, most software rely on existing components. Combining existing components into a new arrangement is an invention process.

What precautions should be taken for such an arrangement? Is it patentable? Can such an arrangement infringe a patent ? Can a source code infringe a patent? Obviously, even if reluctant to it, patentability and infringement issues often could, even should, be considered.

To make and to secure business, some very basic questions could be: What's inside my software? Where does it come from? Is it legally safe to use and combine existing components into new architecture? Could some components be protected by a patent? What implications can this have for the final product distribution scheme?
What about the Internet? Shall a developer consider that since available on the Internet, a component is free [v] of use? Talking about the Internet, never before has it been possible to distribute (did I say copy?) data, files, source code, at such a pace, worldwide, and keeping a track of it. When IP deals with copy rights, no need to explain then the stakes with regard to Intellectual Property Rights (IPRs[vi]) management.

First, let's have a more indepth look into our software.
At first it contains a source code[vii].

<u>A bit of copyright.</u>

Production of a source code is considered as literary works. It is then likely to give right to copyright[viii] protection, as well as the object (compiled) code, but not to the media on which the code is recorded.

Copyright can be considered as a primary mode of legal protection since protection is granted to us immediately, on the date of the creation, since no official filing or registration are required, but provided the creation is *original*[ix]. However, keeping evidence of the creation and of its creation date, through legal deposit is generally recommended since it can be useful in case of litigation.

In addition to the many cases in which authors have the right to be identified on their work, copyright prevents others from:
(a) reproducing (loading, displaying, running, transmitting or storing),
(b) translating, adapting, arranging or doing any other alteration, and
(c) distributing to the public (including renting) the source/object code
without your permission.

The existence of copyright is sometimes sufficient on its own to prevent or stop others from exploiting/trying to exploit your work. If it does not, it gives you the right to take legal action to stop them, and to claim damages.

But as literary works, copyright protection covers the expression of an idea, not the idea itself (the idea itself or rather the functions may be protected by a patent, see below).

Dealing with Software, it means that the code is only protected as a text, an expression, not the underlying idea or functionalities.
Then if anybody wants to develop same functionalities with another source code, there is no way to prevent that, unlike patents.

However, such development shall not copy/translate existing source code. Indeed, also converting a program into a new computer language gives copyright on the new language; it does not abolish existing copyright on the initial existing source code. Converting a program into a new languages counts as adapting' a work. Then re-writing an existing source code into a new language infringes copyright of existing source code.

And if the original source code infringes a patent when compiled and run, translating such source code into another programming language is still an infringement of the patent. With that respect, the strength of patent protection is considered to be globally stronger than copyright.

In addition, copyright applies to any medium. This means that you must not reproduce copyright protected work in another medium without permission.
Indeed, storing any work in a computer is considered as 'copying' the work. Similarly, running a computer program or displaying a work will usually also involves 'copying'. It means for instance that it's not because a code is accessible through the Internet that this code is freely usable: it may be protected by copyright. And downloading such code or embedding it into your own source code could lead to more or less severe issues.

Copyright protection also includes moral rights. Although limited with regard to software, it may be of importance when dealing with authors. The name of the authors should be recorded for every developed element, including when working with sub-contractors, universities, interns, etc.

On the same wavelength, great care should be taken with sub-contractors, otherwise developing software in collaboration e.g. with a university or a start-up company could raise more or less severe issues with regard to the ownership of the copyright when not organised before hand through contract(s).

And this could lead to fundamental valuation issues. What if you can't sign a licence agreement without the consent of a sub-contractor? How would this effect the value of your software?

Another level of complexity may be added when dealing with composite work. Such derivative work occurs when incorporating an existing first work without the collaboration of the author of the first work. This is typical to component-based software and/or to a new release of software. In such cases, the author of the contribution is the owner of its contribution but must respect copyrights of its predecessor(s).

For these reasons, it is better to know and to keep record of who did what, and when. An intern developed an element in a university? A start-up is negotiating a deal with a major company on a software embedding such element? Mr. X -your former colleague- left your company to launch his own product on an object he

developed while at your company? Former friends can become best enemies; and without going that far, having to look for a developer's identity/signature can postpone or cancel a deal.

Why? Because copyright protection grants its owner exclusive rights with regard to reproduction, adaptation and distribution of the work. Copyright can then have power over the distribution scheme. Such legal power is considered as a protective weapon by some and as a constraint by others.

For several reasons, it has been considered that such constraint was an issue, and -to make it short- the open source movement was born.

Open source

Unlike copyright and patents, open source [x] is not ruled by national laws.
Open source is considered as an initiative, a will, a philosophy sometimes, that aims to overcome existing copyright legal constraints though appropriate agreements.

Open source is thus a copyright agreement matter. Several open source licences exist and the reader can refer to Free Software Foundation (FSF) or Open Source Initiative (OSI) for further information.

Among prerogatives of open source software, there is: access to the source code, freedom to modify it, to redistribute copies and to run the software for any purpose.

Open source's seek of legal freedom is a highly powerful tool for developing, adapting and distributing software in that ever changing world, and open source even participates to such changes, as a facilitator for a large distribution as well as a tool to attract new users and potential customers.

Unlike proprietary licence that generally relates to one licensor dealing with one licensee, one of the main features of open source software is the role of a community.

A community gathers users and developers that, for each release, are likely to share experience and to improve software, regardless of their country of residence. A good way to debug and to obsolete yourself before others do…
To be efficient, a community must be active (what's the point of a social network with no events?), and not (at least not too much…) legally restricted.

Then, the licensing of the open source code must be secured, whether you care about others embedding your source code into their own proprietary source code, or not.

One way to secure open source distribution in time (for developments to come) is to impose from the origin a positive copyright constraint. Such positive constraint to copyright is called copyleft.

Copyleft

Whether you like it or not, a source code (open or proprietary) is protected by copyright, although the owner/programmer can decide to enforce it or not. Copyleft is a very good way to control the possibility of distributors to embed your open source code into their own source code.

Indeed, copyleft [xi] ensures that all the authorisations involved in the distribution scheme are systematically granted the same rights on the derived work, which is usually referred to as the "viral scope".
Conversely, the initial author benefits from further developments.

Naturally, reality is a bit more complex and granted rights depend on the type, i.e. the content, of licence agreement. The reader can refer to existing information sources[xii] for further information on open source licences.

Then the type of licence shall be carefully chosen according to your business model, For instance, selling a copyleft licence can sometimes make no sense, and embedding a copylefted element into a proprietary software will strongly and negatively affect the proprietary value of that software.

In addition, it is worth noting that embedding (licensing in) different copylefted components into copylefted software can raise compatibility issues with regard to the "license out" (distribution scheme) that is sought. Basically, it's safer to consider that most licences are likely to be incompatible; and their compatibility must be checked before any licensing out.

Patentability of software

From a technical viewpoint, what is our software doing? How does it work once compiled and run? Does it involve any technical effect on some kind of hardware? Does it achieve any tangible results? If yes, the patent bell should ring because if you could file a patent on this aspect, others could have done it before, or could do it later.

The question here is not to consider whether it's good or bad to have patents on software, but to accept the law as it is[xiii].

In Europe, a patentable invention is a technical solution to a technical problem.

Inventions involving computers have been patentable since the early years of the European Patent Office (EPO) [xiv]. Although a few changes have occurred with regard to how an invention shall be considered in light of the prior art or per se [xv], the law itself has always considered that computer programs "as such" are excluded from patentability[xvi]. Therefore, there must be a clear distinction made between technical (patentable) and non-technical (excluded from patentability) features.

Indeed, with regard to the technical solution aspects, it has been part of the European legal tradition since the early days of the patent system that patent protection should be reserved for technical creations. To be patentable, the subject-matter for which protection is sought must have a "technical character" or involve a "technical teaching", for example, an instruction addressed to a skilled person as to how to solve a particular technical problem (rather than a purely financial, commercial or mathematical problem) using particular technical means [xvii].

In addition, inventions must solve a technical problem. Inventions involving computer programs can be considered as involving technical features, but if they implement business, mathematical or similar methods and do not produce technical effects, they are not patentable because they solve a business problem rather than a technical one. No patents will be granted in Europe for such inventions.

To be patentable an invention must be a computer implemented invention[xviii] (CII), for example, an invention involving hardware shall have a "further technical effect" beyond the normal effects software has [xix], like for instance writing in a memory, communicating on a bus, establishing electrical currents, etc. Similarly, no patent would be granted for an algorithm; however the use of that algorithm in a technical process can be patentable.

Keep in mind that the European law (EPC for European Patent Convention) is a two-edged sword. Not being excluded does not mean the patent is granted. It still has to be new and involve an inventive step, and this is another story…

Then, the EPO does not grant patents for computer programs. The term "software patents" itself is a misleading concept. Only computer implemented invention that make a technical contribution can be patentable. In this respect the granting practice of the EPO differs significantly from that of the United States Patent and Trademark Office (USPTO).

In the USA, only laws of nature, physical phenomena, and abstract ideas are legally excluded from patentability. The Supreme Court confirmed with the *Bilski* case that abstract ideas are not patentable. Business methods and software patents remain potentially eligible for US patent protection. In addition, the Federal Circuit remains free to develop additional criteria for patent eligibility. It means that even if not excluded by law and respecting existing patentability tests, a patent application could possibly face new criteria the Federal Circuit would then design, which creates a bit of uncertainty.

In Asia (China, Japan), the patentability criteria is similar to the EPO, with an even stronger focus on hardware features or “real world” processes.

Then, for those wishing to file patent applications worldwide on software - oops! on a computer implemented invention - there is a kind a paradox in which a "software patent" must be described in terms of hardware. One of the main difficulties is then to transform the software into a computer implemented invention, i.e. at least in Europe to clearly define the technical problem, especially for so-called inventions relating to problems such as a business/administrative method (method of selling, including through a network, data gathering), encryption (DRM), databases and database management system, billing and payment systems, simulations, games, e-learning methods, medical informatics, and mathematical methods [xx].

As Thomas Edison said "The value of an idea lies in the using of it". The same should apply with your software: think of the use of it on its hardware. If the use belongs to one of the previous categories or similar, think twice before spending money on filing a patent application.

In terms of patent application drafting, the Asian criteria being amongst of the toughest, it is recommended to use them as a worldwide reference: if you get a patent granted in China, it is likely to be granted in Europe and in the USA – not conversely, regardless of the Patent Prosecution Highway (PPH) [xxi].

Copyright vs. Patents

Patent and copyright are different legal rights to protect different objects.
Patents cover technical features where copyright protects expressions in texts.

Patents are independent from the distribution scheme of a final product and cannot refer to any open source licence; whereas copyright licences are part of the distribution scheme and can also refer to patent(s).

Patents are examined (what you file is not what you get, and you can get nothing at the end of the day!) where copyright is automatically granted.

Third parties can oppose to a patent where they can't oppose to a copyright.

Basically patents are granted by an Administration (Patent Office) under a public law. Any action from third parties on the patent application or on the patent shall be made before the Administration [xxii] or before national court(s) respectively and will generally deal with the validity of the patent. On the contrary, open source deals with copyright licences ruled under private laws. There is then a possible negotiation on the licence clauses, between the copyright owner and third parties, where the copyright is generally presumed valid.

Depending on the software, securing the investment can sometimes only be made by copyright only, by patents only, or sometimes by both. There is no easy or obvious correlation between protection granted by patent(s) and granted by copyrights on a same product.

As a simple illustration, assume software is protected by copyright only.
If a competitor creates another software with the same functionalities but with another source code, there is nothing you can do about it [xxiii].

But now assume software is protected by a patent.
If a competitor creates software with same functionalities, regardless of the source code, there is a high risk of infringement.

With regard to copyright, protection is granted from the creation. Then, the publication of a source code can be done without interfering with copyright protection.
With regard to patents, the public disclosure of an invention occurs 18 months after the filing date. During that period of time, the patent owner can decide to publish or keep unpublished the content of the invention.

Today, no Patent Office requires or examines source codes nor publishes them as annexes to patent application documents. When filing a computer implemented invention, there is no need to disclose any source code, that is a good way to keep it secret then.

However, any disclosure made available to the public by means of a written or oral description, by use or in any other way, before the date of filing, will destroy the novelty [xxiv] of the invention, then no patent will be granted.

Accordingly, disclosure of a source code can be considered as prior art detrimental to the novelty of a patent application (competitors' ones or your own).

So publishing a source code is a good way to prevent a competitor from filing a patent application on the disclosed technique. However it is also a good way to communicate your knowledge to patent owners.…

So what

Apart from the design of a product, software functionalities are generally what make that product sold.

Software development demands high costs and time investment. In addition, development and distribution are moving processes. New releases are spread in some countries and not in others; software is adapted to some local or to some worldwide client needs. Both in terms of geography, time and technical features, software boundaries are ever changing.

Securing business often requires legal protection, with one or both of the copyright protection and patent protection. There is no philosophical discussion here about whether software inventions deserve patent protection or not.

A patent is very often a vital element for raising funds and for securing access to market, especially for start-ups.

Whereas patents have been extensively used for their "right to exclude", today they are also used as an intangible asset, a valuation tool or a marketing tool of a company; a very good way to gain trust from investors, to raise funds.

Copyright is sometime the only way to protect investment. Copyleft is also a two-edged sword, it can prevent a third party from patenting the disclosed solution, however it also gives access to patentees on your technology.

Patent and copyright (including copyleft) protection should then be considered as part of the strategic business plan of a company, keeping in mind the limits of the respective protection. For instance, since computer programs are not patentable as such, any seizure of a media (CD-ROM, DVD, memory stick, etc) on which is recorded the program will be part of the infringement evidence. Some issue could rise is the device/computer/machine needed to run the program is specifically designed for that purpose/process/software.

In terms of licencing, as a matter of trend, it is interesting to notice that whereas most standard open source licences were quite silent about patents, a few recent licences now include clauses dealing with patents [xxv].

Patents and copyrights can coexist (or not!) on a final product. Existing components to be integrated can be not compatible due to their respective IPRs. Great care should then be taken on each integrated element. In particular, downloading an element from the Internet and integrating it without any consideration to its IPRs can have a strong impact on the final product distribution scheme.

Each embedded component implies its own legal constraints (copyright, copyleft). The *licensing in* policy will impact the *licensing out* strategy, and *vice versa*. The more components the software contains, the more complex is the legal situation.

Open source licences must be carefully checked, at least on their viral scope, their permissibility and compatibility (ascending / descending), before integrating any element. Decision to integrate, rewrite or discard each element should be taken according to a clear licensing in policy and a licensing out strategy.

Talking about strategy, it is interesting to note that some firms develop at the same time version N of a software as open source to attract interest, and version N-+1 as proprietary with improved functionalities. Food for thought isn't it?

As we have seen, that world is an exciting one, not yet fully explored, with moving boundaries.

What to do with all that, then?

Business management / distribution scheme

Dealing with IPRs, a lack of management leads to potentially huge legal and industrial risks that can have a strong impact on the business model and on an economic situation.

Integrating existing components is a "licensing in" process that can be dangerous if programmers ignore the IPRs of others, with a strong impact on the "licensing out" distribution scheme (open source or proprietary).

Each integrated component can be ruled by its own patent(s) and/or its own copyright (copyleft). Integrating elements under a General Public Licence (GPL) for instance can prevent deploying the result as proprietary.

Several elements can be regulated by different copyrights, which are not compatible when integrating them. Integrating patented elements without the consent of the patent owner is an infringement.

Obtaining or negotiating a licence on a patented element can also have an impact on the distribution scheme. Granting a licence on a patent for a proprietary distribution scheme does not implicitly imply a licence for an open source distribution scheme.

As a result, the distribution scheme sometimes has to be modified. The final product must be distributed without one component or with exception; one component must be rewritten or substituted by another one; a negotiation must take place with the IPR owner. Then the time to market is often lengthened and the target price modified.

The choice of using a component is not only a matter of technical features. IPRs of such a component are strongly recommended to be contemplated before any integration. With that respect, the reader is invited to refer to some existing interesting IPR methodologies[xxvi].

The licensing in policy and the licensing out scheme are advantageously linked together in a loop process.

Even for small and medium enterprises (SMEs) choosing the right IPR strategy, i-e copyright, open source or patents as business strategic elements is a way to sustain growth of a company. Nowadays, it seems that sooner or later, most businesses are subject to merger and acquisitions (M&A). With that respect, acting as if you were running a due diligence on yourself is a good way to manage your IPRs.

Securing business involving software is complex and requires a team of specialists.
Among them: developers, technology transfer officers / business development specialists, patent attorneys, lawyers and executives. Naturally, the business model, the legal constraints will be different according to the context. A software developed under collaborative innovation will face different issues than when developed by (or with) a start-up. Producing capabilities of a SME will be balanced by the subcontracting necessities of a University. Similarly, the situation will be different whether software is developed for internal clients within a group or as a product to be launched in the cloud in the e-economy.

In addition to these moving boundaries, it is worth noticing that the EPO trend is to shift the exclusion of computer program from patentability issue to the inventive step requirement. And this is quite another story.

© Thibault Bouvier – September 2012
EP Patent Attorney

---

[i] A **computer program** is a list of computer instructions or data that enable a computer as to execute given tasks/function when compiled in a machine readable and executed on said computer. A computer program can be expressed in two ways: as a source code and as an object code.

> **Software** then will be understood as a computer program, knowing that such definition could also encompass the packaging and the brand under which the software is placed on the market. A brand can be protected by a **trademark**, which relates to distinctive signs, and will not be further discussed.

[ii] **Patents** relate to inventions that involve technical features.

> A patent grants to the owner a temporary exclusive right preventing others -especially competitors- from using the patented invention without his or her consent, in return for public disclosure of the invention. It needs to be filed and generally requires several years of examination proceedings to grant (or not) functional features together with qualified people to interpret the scope of the protection sought.

[iii] **Open source** is a choice on the way to distribute software by giving explicit access to the source code that is protected by **copyright** (see under). Open source software can contain copylefted elements, **copyleft** being the use of copyright law to offer the right to distribute copies and modified versions of a work and requiring that the same rights be preserved in modified versions of the work. Open source then relates to copyright **licensing** and shall not be considered as an IPR (see under).

[iv] See "Patents for software?" document from the EPO, from http://www.epo.org/news-issues/issues/computers/software.html

[v] Please note that the term “free” when associated with open source and software, relates to copyright and is supposed to be well known as clearly differentiated from "free of charge"?

[vi] **IPR** (Intellectual Property Rights) is a term referring to a number of distinct types of creations of the mind for which a set of exclusive rights are recognised. Common types of IPR include patents, copyrights, trademarks, industrial design rights and trade secrets in some jurisdictions (source: http://www.ipo.gov.uk/ipresearch-iprights-sum-201107.pdf).

[vii] A **source code** is a text, a sequence of instructions comprehensible by humans and writen in a computer programming language (like C++ for instance). A source code is then compiled into an object code, i.e. a sequence binary of values (0/1) executable by a machine.

[viii] **Copyright** is a set of exclusive rights granted by a state to the creator of an original work or their assignee for a limited period of time upon disclosure of the work. This includes the right to copy, distribute and adapt the work. http://www.ipo.gov.uk/types/copy.htm

[ix] See the Berne Convention for the Protection of Literary and Artistic Works (http://www.wipo.int/treaties/en/ip/berne/trtdocs_wo001.html)

[x] See http://www.fsf.org/ or http://www.opensource.org/docs/osd / for definition, or www.gnu.org for further information

[xi] **Copyleft** is a form of licensing that describes the practice of using copyright law to offer the right to distribute copies and modified versions of a work and requiring that the same rights be preserved in modified versions of the work.
http://www.gnu.org/copyleft/copyleft.en.html

[xii] See for instance (http://www.gnu.org/licenses/licenses.en.html or in french http://www.inria.fr/institut/strategie/logiciel-libre

[xiii]In addition to national laws, see also http://www.wipo.int/sme/en/documents/software_patents.htm

[xiv] See for instance decision T 208/84 "VICOM" (http://www.epo.org/law-practice/case-law-appeals/recent/t840208ep1.html)

[xv] So called “contribution approach”

[xvi] See Art. 52 EPC (http://www.epo.org/law-practice/legal-texts/html/epc/2010/e/ar52.html)
… shall not be regarded as inventions … programs for computers … as such.

[xvii] Op. Cit. ref iv

[xviii] A **computer-implemented invention** is an invention whose implementation involves the use of a computer, computer network or other programmable apparatus, the invention having one or more features which are realised wholly or partly by means of a computer program

[xix] Such approach has been established in the turning-point Board of Appeal case law decision T 1173/97 (http://www.epo.org/law-practice/case-law-appeals/recent/t971173ep1.html) and confirmed by the Enlarge Board of Appeal decision G3/08 (http://www.epo.org/law-practice/case-law-appeals/recent/g080003ex1.html)

[xx] See for instance "Patent Law for Computer Scientists / Steps to Protect Computer-Implemented Inventions" by Closa, D., Gardiner, A., Giemsa, F., Machek, J. (Springer Edition) - ISBN 978-3-642-05077-0

[xxi] The Patent Prosecution Highway (PPH) is a bunch of bilateral agreements between several patent offices (EPO, USA, Japan, China, etc.). To make it simple, it enables a patent granted in one country to be almost automatically granted in the other country. However, and especially dealing with software invention, due to patent procedures and the application criteria of the PPH, it seems unlikely today that a patent granted in the USA could be granted in China using the benefit of the PPH.

[xxii] See the Observations by third parties and the opposition proceedings before the EPO according to Art. 115 EPC. The reader is also invited to see an interesting initiative made by the peer to patent project in collaboration with the USPTO, UKPTO, IP Australia at http://peertopatent.org/ in which the public can help PTOs to find the information relevant to assessing the claims of pending patent applications.

[xxiii] See case SAS Institute Inc v World Programming Limited before the European Court of Justice (pending) that could lead to a kind of protection of the architecture

[xxiv] See Art. 54 EPC (http://www.epo.org/law-practice/legal-texts/html/epc/2010/e/ar54.html)
… The state of the art shall be held to comprise everything made available to the public by means of a written or oral description, by use, or in any other way, before the date of filing …

[xxv] See for instance GPL V.3, Apache, CeCILL V.2 License, etc.

[xxvi] See for instance the following URL http://www.qualipso.org/IPR_methodology from the INRIA (Public science and technology institution fully dedicated to computational sciences)