M@RSOUIN
Môle Armoricain de Recherche sur la SOciété de l'information et des Usages d'INternet

CAHIER DE RECHERCHE.
Octobre 2005 - Numéro 8-2005

New approaches to intellectual property: from open software to knowledge based industrial activities.

Nicolas Jullien[1], Jean-Benoît Zimmermann[2].

1 : M@rsouin
2 : CNRS / GREQAM et IDEP

Nicolas.Jullien@enst-bretagne.fr
Jean-Benoit.Zimmermann@univ-med.fr

http://www.marsouin.org

http://www.vcharite.univ-mrs.fr/greqam/index_fr.php

RÉSUMÉ.

Nous abordons dans cet article la propriété intellectuelle dans le secteur du logiciel. Nous montrons que ni le droit d'auteur, ni le brevet ne sont parfaitement adaptés aux besoins et spécificités de cette industrie. Le modèle « alternatif » du logiciel libre (open source), basé sur un concept juridique innovant, la licence GPL, prend une importance croissante. Nous en présentons les principales caractéristiques, qui imposent au producteur de rendre public le code source du programme protégé par une licence de ce type, ainsi que celui de toutes les modifications postérieures en cas de redistribution. Cela entraine une modification profonde des stratégies industrielles de valorisation de la propriété intellectuelle, allant vers un plus faible niveau de protection. Nous analysons les conséquences d'un tel mouvement en terme institutionnel et de politique publique. Enfin, nous notons que cette approche apparaît exemplaire pour un nombre croissant d'industries dans le contexte d'une économie basée sur la connaissance.

MOTS CLEFS: LOGICIEL LIBRE, PROTECTION INTELLECTUELLE, ÉCONOMIE DE LA CONNAISSANCE.

ABSTRACT.

We analyze the question of intellectual property in computer software, showing that both copyright and patents do not fit to the specificities and needs of this industry. The alternative model of Open Source Software, based on a very new juridical concept called GPL "General Public License", tends to take a growing importance. We explain its main characteristics, which consist to impose to the producers to disclose the source-code of the concerned programs and of any further improvement if they re-distribute/re-sell it. We show that by doing that it introduces a totally different approach of intellectual property within industrial strategies, based on a weaker intellectual protection. We discuss the consequences of such a movement in intistitutional and public policy terms and we enlarge the approach to understand its exemplariness, in the context of a knowledge based economy, for a growing number of industrial activities.

KEYWORDS: OPEN SOURCE SOFTWARE, INTELLECTUAL PROPERTY PROTECTION, THE ECONOMY OF KNOWLEDGE.

1. Introduction.

Though software intellectual property could not satisfactorily fall into any existing legal framework, all countries have taken the decision to range it under the category of copyright. Then the double objective of intellectual property protection is not satisfied, which consists, on the one hand, to grant to the inventor a provisional monopoly for exploiting his invention and, on the other hand, to oblige him to disclose the principles of his invention. To resort to the patents system as it became more and more usual in the US and which is in debate in Europe, raise other kind of problems. Now the alternative model of Open Source Software, based on a very peculiar juridical tool called GPL "General Public Licence", tends to take a growing importance. Its main principle is to impose to its adopters to disclose the source-code of the concerned programs and of any further improvement if they circulate them, as well as the free circulation of the code under the sole condition to maintain its "open" character.
This does not exclude a possible commercialization of these programs and do not limit “open source software” to a non-marketable sphere. Understanding that, firms have more recently joined the world of cooperative development and of free access to sources-codes. This enlargement is concerned with two types of strategies. On the one hand firms distribute open software products to enlarge base of users with the services that can help to use them : training, adaptation to specific case or context, hotline, maintenance, updating, … On the other hand, a growing number of enterprises began to "free" part of their software products aiming to draw benefits from the potential of development of the free software community or to favour a large diffusion of a key-product imposing it as a de facto standard and looking for gains from the commercialisation of proprietary complementary products.

By doing that they introduce a totally different approach of intellectual property within their industrial strategies. But these strategies didn’t accommodate very well with the GPL terms as they were and this actors enlargement has led to a juridical enlargement beyond the strict framework of the GPL. Number of “hybrid” licenses have been designed in order to control the extent of their openness.

So open software approach doesn’t represent a denial of intellectual property but a new way to manage intellectual property. Through the GPL, intellectual property is not rejected, authors do not renounce to their rights but to the sole monopoly rent, such rights would authorize in a copyright regime. The main legal aspect is that, when a program is declared under GPL license, any code derived from it or integrating GPL code lines must also be available under GPL License. Hence GPL status is “contagious” in the sense that this status attached to any number of lines is automatically transmitted to the whole program into which they are incorporated. The authors do authorize anyone who wants to make use of their work (modifications, improvements, additional features …) under the sole condition that the new product could also circulate freely.

Such an approach appears exemplary in the context of a knowledge based economy, for a growing number of industrial activities, for which the scope of knowledge that has to be mastered appears to be too large for a single even powerful agent. Consider knowledge as a mutual resource implies a reshaping of the value chain concept, cash flow being drawn from the usage of the knowledge base (services, complementary products), not from the knowledge itself. We shall illustrate this idea on a short list of knowledge based activities, like biotechnologies and health in order to bring to openness approach a larger extent.

In this paper, we propose to explain the reasons why the copyright framework has been chosen for computer intellectual property protection, why it does not work very well, but also why a patent system may not be better suited (part 1). We will then consider the alternative model of Open Source Software, based on a new concept of authorship rights as expressed by the GPL "General Public License". We shall replace the

appearance of the free-software movement in its peculiar context and explain the main reasons of its present industrial success (part 2 and 3). We will defend the idea that this new way of considering intellectual property management echoes debated in other industries, such as biotechnologies (part 4). This has different implications in institutional and public policy we discuss in the last part (part 5).

2. The nature of computer software and the question of intellectual property.

At the very beginning of computer industry, the question of software protection was not raised in so far as software appeared as bearing a program logic, substitute of the wired logic of hardware, that permit a bigger flexibility to the machines and gave them the ability to process a plurality of tasks onto the same architecture. With the further emergence of a clear distinction between system and application software, the computer became a universal machine able to be dedicated to any application in the sole limits of its own processing performances. But as long as application software was developed and supplied by the computer manufacturers themselves, computer programs have been considered as joint products whose supply was an integrant part of the marketing arguments for selling a given system.

Two important changes have occurred during the seventies and deepened in the eighties, that have assigned to software its proper status. The first was the idea of compatibility, due to Gene Amdhal, supervisor of the IBM 360 series, according to what the software system of a machine could be implemented on a quite different but appropriately architectured machine. From this "transportability", operating systems had to be considered as distinct goods likely to be appropriated by other actors than those who conceived and developed them. This disconnection between hardware and software components of a system architecture has been reinforced with the take-off of microcomputers during the first eighties, when IBM charged an external company, called Microsoft, to develop and to supply, under its own licence, the operating system of its brand new "Personal Computer" . The second change was related to application software. It is the emergence of software houses in the same seventies, as a consequence of the strategy of DEC to supply "mini-computers" as pure data processing systems without any application program, instead of the former bundling offer of larger computer manufacturers. Computers users could then alternatively develop these programs internally from their own skills or sell them from specialized enterprises. By the time, the first custom-made programs gave rise to a new generation of standardised portable products able to be implemented on a large range of machines at a very low marginal cost.

But the most influential event that led to consider software as a market good has probably been, as early as the end of the sixties, the decision of US justice to impose the unbundling of software-hardware IBM's supply, hence a separate billing of software components. This was the conclusion given to the American federal then the European suits against IBM in the name of anti-trust law and for dominant position abuse. By the way software products had to reveal their market value and to compete with those from other computer suppliers or independent software developers.

From these interdependent dimensions of historical evolution, that gave to software a position of market good, stemmed the need to determine a framework for its intellectual property recognition and protection. The early studies devoted to this question in Europe as well as in the United States during the seventies recognized that not any of the existing protection frameworks could be considered as totally satisfactory. But they recommended to avoid a long delay required for a new framework design and general adoption at the international level and the risk of a rapid obsolescence as a consequence of a quick and deep technological change (See OTA, 1992). For those reasons, most of the developed countries decided to ad-

opt the copyright law as a common reference for software intellectual property protection, with several variants resulting from national juridical contexts.

Actually the question of software intellectual property protection is not quite satisfactorily settled by the copyright protection owing to the very specific nature of the software good and its production conditions. First of all a software product can be considered as an intellectual expression of ideas that is coded by the use of a specific programming language, with its proper vocabulary, syntax and structural rules. For such a reason its protection has been considered as falling in the field of copyright. But from a practical point of view, a software program aims to carry out a given task leaning on the resources of the computer it is implemented in or, in the case of a system software, to coordinate the running of the different components of the computer architecture. For this purpose a software product will be "translated" from its explicit expression, called the "source-code", in a given programming language, to a new form directly "understandable" by the machine and very far from human understanding. This new form, obtained through a "compilation" operation, is called the "object-code"; it is the same program but its initial expression is not anymore readable. If the source-code is not supplied jointly it can only be imperfectly and costly restored through a heavy operation of reverse engineering. Implemented on a machine, the program is then able to emulate properly its resources for a given task without requiring from the user a precise knowledge of the technical process that is set to work. In that sense it is a technology and should fall in the field of patents.

The basic principle of intellectual property protection is to bring an acceptable compromise between granting incentives to the inventor through temporary monopoly rights on the commercial exploitation of his invention and favouring the diffusion of knowledge by compelling him to disclose the principles of his invention. In the option of software protection through copyright, the problem is that copyright protects a given expression on the ideas and not the ideas themselves. Then software producers are not obliged to disclose the source-code of the protected programs. Most of the editors do commercialise their software products in the sole form of executable programs. They generally do not reveal the "source-code" of the programs, that is the explicit expression of the program architecture, procedures and algorithms. This appears totally contradictory with the aims of intellectual property protection in so far as the owner of intellectual property is not at all constrained to reveal any information on the working principles of the protected program. The American jurisprudence has adopted a quite severe attitude on this concern, strengthening the protection, by condemning for copyright infringement any suspected attempt of reverse engineering on copyrighted programs.

But such a standpoint doesn't give any satisfactory answer to the needs for compatibility and interoperability of the different programs likely to be implemented on a given computer. The general European position on this issue has been to oblige the editors to reveal the interface specifications of their programs (the formats and protocols required for exchanging data with them) or to permit a reverse engineering operation in the sole aim of disclosing these interface specifications. This position has progressively induced editors to publish spontaneously their interfaces, avoiding by the same way to be prosecuted for anticompetitive practices. An alternative solution to this question could be reached by substituting public interface standards to those private specifications. The argument is that interfaces specification do not correspond to any inventive activity but depends from arbitrary options. Their social value is the sole result of a collective process of adoption. A standardisation process appears then as a co-ordination tool aiming the conciliation between individual private preferences of firms and collective choices issued through committees or standardisation bodies (Farrell and Saloner, 1988). The archetypical illustration is the attempt supported by ISO and IEEE[1] in the late eighties to set up

[1] ISO : International Standards Organization ; IEEE : Institute of Electrical and Electronic Ingeneers.

an *Open Systems* approach in the framework of Unix based computers, aiming the definition non-proprietary interface standards while preserving the variety of proprietary designed architectures (Saloner, 1990). Unfortunately this approach collapsed because of the supremacy of firms strategies aiming the building and preservation of dominant market positions.

In spite of the juridical preference for copyright law expressed in the seventies, more and more numerous demands of patents have been granted, since the nineties in the Unites States, for software programs or even simple procedures or algorithms. In Europe, the European Office of Patents remained on its initial position to grant patents in the sole case they are integrant component of a industrial device or process. But, more recently the European commission has submitted to the debate a new directive aiming the patentability of software. This evolution can have very important consequences in terms of software industry structure and innovation dynamics. On the one hand, a large part of algorithms and procedures that programmers make use all along their development work has been considered until now as belonging to the public domain, then being freely available. In the absence of a real state-of-art in the field of computer software programming, patents are granted on a totally arbitrary way, giving private rights for the use of resources that had been formerly shared by professional without any reference to their origin. On the other hand a generalization of the patent system would imply a progressive partitioning of knowledge and practices in a domain where innovation is based on cumulativeness and complementarities. In such conditions, a strong regime of intellectual property protection would have dramatic consequences on the dynamics of innovation (Bessen and Makins, 2000). "Entry competition and innovation may be easier if a competitor needs only to produce a single better component, which can then hook up the market range of complementary components, than if each innovator must develop an entire *system*" (Farrell, 1989).

This problem appears particularly crucial regarding that more and more complex software products have been designed thanks to modern structural programming methods. Programs are built from the combination of elementary modules into a global architecture. This approach goes, on the one hand, with a increasing recourse to a large scope of software components, portable and reusable in different contexts, and, on the other hand, with a growing proximity with the mathematical foundations of programming. This evolution raises more accurately the problem of the distinction between public and private property of modules and algorithms. Copyright laws have rejected principles and algorithms from the scope of protection. But patents granting for software components creates a barrier to their usage and contradicts the way of working of the whole community of software developers. The sole actors that will be able to manage such a situation are the large companies that will have the capacity to build a large portfolio of patents. For SMEs that will be attacked for patent infringement, whether it is effective or not, the cost for the defence will be so hight that it will threaten their survival.[2] And, as a matter of fact, the champions for the constitution of patents portfolios during the last ten years are not only software editors but also firms, newcomers in the information technology field, that understood the opportunity to build, at low cost, the basic material for future speculative profits[3]… It is also a real threat for open source software as illustrated by the SCO case. SCO Group born from the merger of Caldera Systems and Santa Cruz Operations asserts the ownership of part of the Unix codes used in the Linux kernel. It claims for that one billion dollars to IBM and send a letter to 1500 other companies to inform them of the risk they run in the

[2] See « Brevets logiciels et Linux : des chiffres qui inquiètent », http://www.journalinformatique.com/0408/040803_linux.shtml

[3] It is the case of the US company Acacia, formerly start-ups incubator whose sole activity is right now to sell licenses under the threat of lawsuits. See A.Chassignin, « Ces sociétés qui tirent profit des brevets logiciels » , http://solutions.journaldunet.com/0409/040906_brevets.shtml

case they continue to offer solutions derived from GNU/Linux[4].

This progressive but inescapable evolution reveals a fundamental conflict between two opposite conceptions of software development and innovation, depending on whether the core resource of the activity is to be found in the creative potential of developers teams or in the monopoly power of the firm that employs them. And the basic distinction with traditional industrial activities is that, this time, the main input of the production process is of informational and cognitive nature.

This debate is all the more on the agenda so as a growing part of software production is distributed under "open" licenses, making this production freely appropriable. Thanks to the Internet, developers belonging to different institutions collaborate to develop such software. We will detail the origin of this movement and the reasons if its success in the following parts.

The Free Software initiative: context and goals.

By the end of the eighties, the main production was done by the private sector and the industry increasingly considered software pieces as products rather than the outcome of a service. Even though, following the long tradition of public research in the USA, the universities continued to disseminate their production with very liberal licenses, such as the BSD[5] license.

But this system had perverse effects, especially for IT professionals. As they did not have access to the source code, they could not adapt the programs to specific configurations or to specific needs. This is the origin of the Free Software Movement, initiated by Richard Stallman, quite a famous researcher at MIT at that time[6].

His goal was the following: to develop a whole free operating system (ie the kernel, but also development tools, (compilers, programming tools), or graphic user interface, etc.) In the mind of its creator, "free" means a broader spectrum of "rights"[7]:

- "the freedom to run the program, for any purpose (freedom 0),
- the freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this,
- the freedom to redistribute copies so you can help your neighbor (freedom 2),
- the freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this."

A new way of managing proprietary rights.

As it is stressed in these lines, the main difference with private licenses is the access to the way the program has been created (the source code) and the freedom to modify this source code. There is no difference with the existing free licenses, such as BSD which respects these points.

The main innovation lies in the creation of a specific license, the GPL[8], which grants the "rights" quoted, but which provides the devel-

[4] See for example Estelle Dumout, « Le camp des logiciels libres dénonce SCO dans sa guerre contre Linux » http://www.zdnet.fr/actualites/informatique/0,39040745,2135115,00.htm

[5] For "Berkeley Software Distribution" license. This license allows anybody to reuse the program, to modify it and to redistribute it under the terms he or she wants. The argument being that this production has been funded by the public and should be available to the public. SUN (for Stanford University Network) used Berkeley Unix and Stanford network system to construct its first commercial offers.

[6] He has created, among other things, and as the first Free software, Emacs editor (1984), still used and among the best program design tools today, and the GCC C and C++ (among other languages) compiler (1985), the most used today.

[7] Taken from the Free Software Organization, at http://www.gnu.org/philosophy/free-sw.html

opers with a new tool, the fact that if somebody wants to redistribute a program under GPL, with or without modifications he or she has made, this redistribution must be done under the same terms and conditions. The will is to ensure that the program will not be closed, as BSD protected programs can.

To be coherent with this point Stallman resigned from MIT to create a foundation (the Free Software Fundation, or FSF) in charge of hiring people to develop the Free operating system, since as an MIT employee his production would have been owned by MIT.

This shows that the Free software movement, is, since the beginning, very aware of intellectual property rights. GPL does not represent a denial of intellectual property but a new way to manage this intellectual property. Through GPL-like licenses, intellectual property is not rejected, authors do not renounce their rights but choose to distribute and to keep open their intellectual production. This original way of managing intellectual property has been called by its initiators the "CopyLeft" attitude (once again mirroring the juridical reference, the copyright).

This action is to be analyzed in the specific professional context of software production.

A specific professional context.

Stallman has presented his arguments for initiating this project[9]. Our analysis of these arguments is the following:

- **this occurred in a specific professional culture context**[10]: computer research centers, which were used to collaborate in the US (private and public ones), since the beginning of computer history, in particular in the exchange of program files. Computer science courses are partly based on the principle that it is more efficient to reuse what exists than to redevelop from scratch. This has been "codified" in the hacker philosophy by Eric Raymond, among others[11]. Mainly concentrated into research centers (again, public and private ones, such as IBM or Xerox), this professional culture diffused itself at the same time that students in computer science were hired by firms,
- thus, technically, in terms of productivity, the closure of private programs and the closure of public programs (such as BSD protected Berkeley Unix) under private distribution where perceived as very inefficient for this category of developers. For the first time they lost control of their working tools. This triggered the reaction of creating open, "free" programs and a juridical tool granting that these programs should remain free,

In addition to this, it must be clear that, also in this context, history mattered:

- **a personality**, **Richard Stallman,** has made the difference between a vague feeling of resentment towards the closure of programs in the IT professional community and the construction of a coordinated "riposte". He has convictions (in the debate of who has to produce a public good, Stallman stands that this public good must remain public as research production, being intellectual production), and he has agreed to renounce its professional situation to defend them. And he had the charisma to be respectfully heard by his "co-developers",
- **new technical tools, especially the Internet**, have made this campaign possible and have largely facilitated the diffusion of FSF production (via "mailing" lists and ftp sites).

[8] General Public License, which protects programs like GNU/ Linux.

See: http://www.gnu.org/licenses/licenses.html#TOCGPL

[9] http://www.gnu.org/gnu/thegnuproject.html

[10] We have developed this argument in Jullien [1999].

[11] http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/. Hacker being understood in its original acceptation, i.e. high skilled developer.

Finally the diffusion of the Internet, in non-US research centers first and to the public later, has increased the number of users and developers of Free software. Internet tools have made possible the co-development of making this movement a new organizational system to produce software.

The consequence: a new way of producing software.

Internet allows developers to follow the life of a project, reading mailing lists, downloading software and documentation. It has also made it possible for different developers, localized in different places to co-develop the same software project.

This organization, based on voluntary contributions, has been described first by Raymond [1999]. Each project is led by a core group of developers (the 'kernel') which develops the majority of the source code[12]. A larger group follows the development, sometimes reports bugs, proposes corrections ('patches') or new developments. And an even larger group just uses the program and sometime posts some questions on the use of this program on user mailing lists. If, in details, the organizations differ from one project to another, there are always mechanisms to select the contributions, the questions, so the main developers should not be inundated by peripheral problems[13]. Another very important characteristic is that big projects are split in coordinated "small", easier to manage sub-projects.

As explained in Demazière et al. [2004] and in Jullien [2001], individual motivations for initiating a Free software project or for collaborating to an existing project are numerous:

- for high skilled developers, it costs sometimes less to develop a program from scratch than to use one which does not exactly meets its needs. Once developed, a free publication allows a quicker diffusion and thus quicker feed backs, very useful to track bugs and to improve the functionalities of the program,
- following the same principle, which guided Stallman's initiative, such developers find very interesting to use open source software to be able to adapt them to their needs. The return of the bug found, the correction of those bugs or the modifications, once done, does not cost very much. And as for freeing a program, the developer is granted that this problem solution or program modification will be integrated and maintained in the next generations.

It is clear that carrier concerns, pointed out by Lerner and Tirole [2002], or the social capital a hacker can earn maintaining or contributing to a much used free software is also a factor for keeping these persons motivated with this task. However, this does not seem to have been anticipated by the developers we have interviewed (Demazière et al. [2004]), when they started to contribute to a free software project. But this gives a key to understanding the rather more surprising involvement of companies in free software development, which will be analyzed in the following part.

AN INDUSTRIAL SUCCESS HISTORICALLY SITUATED.

The situation.

Today, since the anouncement made by IBM in 2001of investing $1b in Linux[14] development, free software has been adopted in many commercial offers. Novell, buying Ximian and SuSE

[12] See for instance Mocus & al. [2000] for a presentation of Apache development organization.

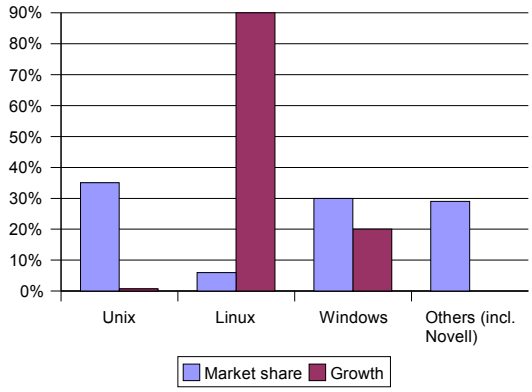
[13] This can be a problem appearing with the success of a program. If there were not such mechanism, the most productive developers would progressively dedicate more and more time to addressing basic problems and thus losing interest in the development until quitting. See Foray and Zimmermann [2001] for a discussion on this point. For a description of the organization, see Demazière et al. [2004].

[14] For which the return has been in no more than a year, according to the company, see http://news.com.com/2100-1001-825723.html

has built a new libre software based strategy, Sun has annouced it will open-source license its operating system, Solaris[15]. IBM has turned its developement tool software Eclipse into open-source. Even Microsoft has released some programs under open-source license[16].

In terms of business, because of a very high growth rate, Linux generates significant business (see Figure 1).

Figure 1 Global market share and evolution for operating systems in 2003 (in value, source IDC & Gartner).



IDC, quoted by KBC Securites in a financial analysis of the Linux distribution producer MandrakeSoft[17], forecasts a market share of 33% for Linux in 2007 in volume.

All these studies are centered on the key element in a computer system, the operating system. But in a growing number of markets, open source software products are very present: PHP is the leading language for the production of dynamic Web sites, Open Office is today a credible alternative to Microsoft Office, gaining important clients, such as the Ontario Ministry of Education[18]. MySQL data base software is also increasingly used, particularly in dynamic Web site design. The Spanish region of Extremadura has even based its IT development policy on libre software, editing its own distribution of libre software[19].

Even if it can appear at odds with the 'software package' culture, this success can be explained.

An analysis.

A first explanation.

First, the actual success is historically situated. Stallman initiated his program in 1984, but the appearance of free software in commercial offers can be dated from the end of the nineties. This is mainly due to the diffusion of Internet tools, which have been developed in US universities/European Research centers and distributed as 'free' software (Sendmail server, HTML language, NCSA Web server, etc.) At the beginning of the diffusion of the Internet into organizations (firms, administrations), servers were installed by engineers who had discovered these tools at university when they were students. As, in addition, they often did not have any budgets, they installed what they knew at the lowest cost: 'free' software. We can consider that the installed base in universities has initiated a standardization effect, or an ``increasing return to adoption"[20], to use Arthur [1988], [1989]'s terms.

However this does not explain the never-failing diffusion of such software, outside the Internet sphere and outside low-cost, low-budget projects, which is today very impressive[21], nor the financial reward system which should repay

[15] http://www.theregister.co.uk/2005/01/19/sun_cddl_solaris/

[16] http://solutions.journaldunet.com/0404/040407_microsoft.shtml

[17] http://www.mandrakesoft.com/toprint/KBC-Analyse-MDK-280604-en.pdf

[18] http://news.zdnet.com/2100-3513_22-5227983.html?tag=zdnn.alert

[19] See http://www.linex.org/ and http://www.linex.org/linex2/linex/ingles/index_ing.html for the reasons for such a political choice.

[20] Which means that each adoption reinforces the incentive for an adopter to adopt himself.

[21] See, for instance, Netcraft Web server survey (http://news.netcraft.com/archives/web_server_survey.html), or the fact that today Linux appears as the competitor for Microsoft's Windows family, not only for analysts, but also for Microsoft. For the first time in its history, this company is doing comparative advertising between its products and Linux (http://www.microsoft.com/mscorp/facts/default.asp). The last success story being the launch of firefox, which, in few weeks, has perceptibly caused to crumble Internet Explorer's dominant position in the browser market.

those who have invested in open-source software development and diffusion. This is based on the mere characteristics of these open systems, which propose solutions of very good quality at a very low cost to professionals.

The quality of the software produced.

More than mere public research products, free, open-source programs were, first and foremost, tools developed by user-experts, to meet their own needs. These user-experts are behind many software development initiatives (among which are Linux, Apache or Samba) and still improve them. And the organization[22] or production has obtained remarkable results in term of quality and speed of improvement[23]. A recent report by the Coverity company states that "Linux is a very good system in terms of bug density" and may have many times fewer bugs than typical commercial software[24].

It is undoubtedly due to the free availability of the sources. This allowed skilled users to test the software programs, to study their code and correct it if they found errors. The higher the number of contributors, the greater the chance that one of them will find an error, and will know how to correct it. Programming rules are also very strict to make this reading possible[25]. All this contributes to guaranteed minimum thresholds of robustness for the software.

Co-operative work, the fact that the software programs are often a collection of simultaneously evolving small-scale projects, also require that communication interfaces should be made public and "normalized"[26]. Open codes do facilitate the checking of this compatibility and, if need be, the modification of the interfaces. It is also remarkable to note that, in order to avoid the reproduction of diverging versions of Unix, computer firms have set up organizations which must guarantee the compatibility of the various versions and distribution of Linux[27].

All this led Zimmermann, [1999] to say that, "if the solution of the open systems has suffered because of a recent return to the more isolated proprietary systems, the solution offered by open-source programs could contribute to go beyond the limits and contradictions of the systems of intellectual property. It might even lead to a thorough reconfiguration of the software industry". The use of such programs by companies can be seen as the creation of professional tools to collectively coordinate to create components and software program bricks which are both reliable and, again, "normalized". Up to now, this collective, normalized base has been lacking within the information technology industry (Dréan, [1996]).

But this organization works with software literate people and companies, able to define their needs in terms of software specifications, to evaluate a program download, to install it and hack it if needed, just as for classical "private" programs. The majority of the users, being firms, administrations or even single users are not always competent to do so. That is why,

[22] For more about the way open development is structured, besides Raymond, [1998a], one can also refer to Lakhani and von Hippel, [2003] and Jullien, [2001].

[23] See Tzu-Ying and Jen-Fang, [2004] for a survey and an analysis of on-line user community involvement efficiency, Bessen, [2002] and Baldwin and Clark, [2003] for a theoretical analysis of the impact of open-source software code architecture on the efficiency of open-source development. The latter argue that it may be seen as a new development "institution" (p. 35 and later) .

As for performance tests, one can refer to http://gnet.dhs.org/stories/bloor.php3 for operating systems. The results for numerous comparative evaluations are available on the following sites http://www.spec.org and http://www.kegel.com/nt-linux-benchmarks.html (the latter mainly deals with NT/Linux).

[24] See the December 13 CNet article by Robert Lemos, http://news.zdnet.com/2100-1009_22-5489804.html.

[25] Let us remember that the first GPL programs that were created and adopted were development tools (in particular, compilers, interpreters and programming assistance tools). In fact, software programs allowed the normalization of computer languages, which are the minimal common base necessary to communicate. See http://www.gnu.org/prep/standards.html for technical recommendations on how to program GNU software.

[26] In the sense that they respect public formats whose evolution is decided collectively.

[27] It is the Free Standard Group (http://www.freestandards.org/). Are, among others, members of this committee: Red Hat, Mandrake, SuSE, VA Software, Turbo Linux, but also IBM, SUN or Dell.

since long time ago, some people from open-source movement argue that “companies should be created and that this activity should be profitable” (Ousterhout, [1999]). This is the origin of the first companies which have based their business on free, GPL protected software.

The business.

In fact, we can distinguish two periods, and two kinds of actors in the business development: the newcomers period, with new firms coming with these new programs, trying to propose a new business relationship, and the incumbent present period, with firms seeing the success of such products and trying either to integrate these products into their offers, or to adapt the open source concept to their business. The global result is a deep evolution of the market.

The first business was to address individual people or companies wanting to test or install Linux or other open-source software, in a context where Internet broadband for individual connection was expensive. Two new types of firm appeared:

- ‘distribution editors’, like RedHat, Mandrakesoft, or SuSE proposed the assembling of multiple software programs around one operating system, and developed tools to install and manage these programs. They sold this in CD-Rom, through the usual networks (bookstores, IT retailers). But, today, it is not profitable to produce and distribute standard GNU/Linux distributions through this network. The limits of such a positioning lie in the fact that, since these distributors use free products, they can encounter competition since similar products can distributed by other producers. Free software does not constitute a source of income but is simply a loss leader for maintenance services. Today, these companies hope to develop resources by proposing services with added value, thus increasing their "ARPU".[28] Doing so, they compete with service companies…
- service companies, like Linagora in France, propose solutions based on open-source software to organizations. They see their competitive advantage in the quality of the open-source software, the fact they are easier to adapt and to maintain. Producers can more easily guarantee, through a contract, the reliability of the programs they use, as they can intervene themselves on these software programs. In addition, when programs are GPL protected, the fact that the software program sources are accessible and that the evolution of these programs is not controlled by a firm, can reassure the adopter: the solution conforms and will continue to conform to the standards. It will thus remain inter-operable with the other programs he/she uses. But today these firms compete with traditional service companies, or constructors such as IBM, which use the same open software and propose the same services.

For the time being, newcomers have not acquired dominant positions in the computer market thanks to these new ways of producing software. But this is more due to the successful diffusion of the use of open-source software in business than a lack of success of these products. Newcomers face tough competition because incumbents have adopted these open source products, contributing to their development, either participating in existing open-source projects, or opening their own production.

They participate in existing projects, especially in Internet software (Apache, Sendmail) and operating systems like Linux first for opportunistic reasons:

- some of these programs are dominant in the Internet market. These firms need to contribute to them, to make sure that the programs they produced are well taken into account by the standard, if they want to access the market[29,]

[28] "Average Revenue Per User". This term is mainly used in telecommunications and makes it possible to evaluate the profitability of a firm by basing oneself on the average income generated by a user.

[29] The best example of this type of behavior is undoubtedly the work that SUN has done to develop the Apache free software program. This firm deals with everything that relates to the adaptation of Java to Apache. By making the language it has developed compatible with the market

- in some markets, challengers see opportunities to sponsor a standard in competition with the dominating standard. This is the case, for instance in operating systems: some competitors (IBM, HP, Novell now), of SUN in the Unix market, or of Microsoft in the operating systems for the servers' market, have strong incentives to support Linux. It creates a competitor to these firms at a very low cost, and being GPL protected, this program cannot be appropriated by a single firm, which would mean coming back to the situation they fight using Linux.

GPL protected software, when standardization effects are important appears to be a way to solve the "wedding game" situation: actors want to impose their view but less than to see a competitor imposing its own. So when you are not the leader, it can be interesting to favor an open, non 'privatizable' solution. Such 'open' organization allows such creation of "public industrial goods" (to refer to Romer [1993]).

But, more interesting in the long run for the sustainability of the model of production, traditional software producers have also understood the interest of this open-source based collective development. Opening the sources has several advantages:

– it allows better feed-back from users, as they can propose and develop new functionalities by themselves, better bug-tracking, as they can report some bugs. In a word, it is a powerful means to externalize a part of the development costs[30],

– it is also a good signal to show that the component produced respects norms and standard, especially for the interfaces, as the program code is public.

These software producers financed their development costs selling the product. With GPL like licenses, this results difficult, but not impossible. Some have successfully made the business evolve to service (assistance for the use of the program, maintenance contracts, specific development to adapt the product to client's needs). Some examples of such adaptations are Zope corp. or Adacore companies[31].

Other are trying to keep the advantage of both worlds with specific licenses: the externalization of R&D, as people can have access to the source code, propose some modifications, and the financing system, as the original owner is the only one authorized to integrate and redistribute these modifications and/or to sell the product. For commercial use, some licenses hold that a fee must be paid to the original owner (SUN or Microsoft licenses are examples of that system). Others have double licensing systems, one allowing free use and access to the source code for non-commercial purpose, the second requiring a fee and restricting modification in commercial situations (this was MySQL strategy for instance).

Thanks to this innovative way of considering the licensing tool, companies have today a portfolio of strategies to valorize their intellectual property (see Muselli [2002] for an analysis of the scope of these strategies), even if we doubt the efficiency of these open but not completely free licenses since users may not trust the producer in its will to keep open the sources and are more reluctant to cooperate.

The pooling of open bricks should also change the competition on service towards long-term relationships and maintenance services for software programs, instead of the basic installation of private programs. We defend the idea (Jullien 2002) that this can encourage firms to improve customer services, which is one of the weaknesses of the computer industry[32].

standard, it can hope to sell the Java development tools which it produces (as well as its expertise in this field).

[30] See, the Microsoft 'shared source program' on http://www.microsoft.com/resources/sharedsource/

[31] Zope corp. edits Zope Content management softawe (http://www.zope.org). Adacore edits the Ada 95 compiler named GNAT (http://www.adacore.com). Ada 95 is a programming language designed for large, long-lived applications - and embedded systems in particular - where reliability and efficiency are essential.

[32] See on that weakness the analysis by de Bandt (1995), or Dréan (1996) (p. 276 and following).

3.A NEW WAY OF MANAGING INTELLECTUAL PROPERTY AND PRODUCTION IN KNOWLEDGE INTENSIVE ACTIVITIES.

In any case open source principles can be considered as a rejection of intellectual property. On the contrary to that paradoxical application of copyright law to software, open source gives a better place to authorship. Protection of software by copyright creates a situation where private property and secrecy are reconciled in a total opposition with the foundations of authorship protection (Vivant, 1993). In contrast, open source includes the identification of individual contributors for any part of a given program or further improvements or modifications. On the one hand, this takes part to the building of each contributor's individual reputation and play a role as individual incentives to contribute. On the other hand, the signature of the individual origin of each part of the program, is a guarantee of the seriousness of individual contributions. By this way, individual identity is not dissolved in a collective purpose and the literacy dimension of the code remains accessible.

By connecting a large amount of individuals around a common project and without any closure, open software is a way to take advantage of a fantastic potential of distributed skills. And by return developers not only expect the benefits they will draw from the availability of the public good that constitutes the open source program, but also will benefit from individual learning effects (Foray and Zimmermann, 2001). Due to the open context of working, learning through contributing to a free software project can be considered as more efficient than in the boundaries of a close enterprise. This can be explained by the multiple interactions a developer benefits with a wide variety of programmers using a wide scope of methods and programming styles. This "learning by interacting" represents an other complementary dimension of individual incentives. Such variety is also a source of quality for programming goals.

But, to do so, the functioning of an open source community has to satisfy to two different levels of constraints that could ensure at the same time decentralisation and coherence of the project advancement. First of all, as any complex software project in the classical context of production, the program architecture has to be conceived in terms of modularity and structural sequences. This will permit programmers to work on a module development without necessarily be aware of the detailed functioning aspects of the other components that can be considered as black-boxes. Then the development can be shared on a distributed division of labour and successive steps of the code can be produced either sequentially or simultaneously. This constraint is stronger enough for having made necessary to redesign (and redevelop) some programs as well known as Apache. Being that, the open source development mode generally doesn't involve a fixed and pre-established division of labour among the developers. Demands for contributions to components production or improvement are usually sent out towards a broad and open population of developers in order to improve the chance to get a good and fitted solution and to benefit for that from a large variety of skills and approaches. In addition, improvements or modifications can be proposed spontaneously and developers often already have on their own shelves the answer to the raised question (Von Hippel, 2001). Then it is the responsibility of a core group of developers to establish the relevant orientations of the development, select the contributions, articulate them in the global architecture of the program and to supervise a large part of the developments involved in the program structure[33]. This is the condition of coherence and quality of the collective effort, it is an essential task of coordination of the community production.

But for doing so, such a core group needs to be devoid of private economic concern into the project. That's the reason why individuals of the core group are chosen from the sole considera-

[33] Following Mocus and al. (2000) about the Apache project "despite broad overall participation in the project, almost all new functionality is implemented and maintained by the core group" (p.268).

tion of their technical competences and are co-opted by the developers group alike in the scientific community. In this way the developers' community preserve all the characteristics and merits of an epistemic community: cognitive goal, commonly accepted structural authority, synergy of individual variety, individuals' knowledge accumulation based on their own experiences, recruitment with regard to agent's contribution to the common goal (Amin et Cohendet, 2004). From the other side the way of working of open source production, that is allowed by the principles of open intellectual property management, endows the community with the same coordination efficiency, in terms of collective production consistency, than a formal organisation but while preserving its flexibility and the wide extent of its resources. It forms a very motivating mode of decentralised coordination very similar to firms cooperation networks but more flexible and never entangled by intellectual property conflicts.

Demazière, Horn and Jullien (2004) use the expression of "distant community" referring to the fact that the individuals involved are not only dispersed in geographical or organisational terms but also to their heterogeneity regarding their individual profiles. First developers can occupy various professional positions : students, universities or research centres employees, workforce of private enterprises linked to open source software, personnel of other enterprises… For those different categories there is a different balance between open source development activity and remunerated work, inducing a plurality of the juridical and temporal conditions of the contributions. This statutory variety has to be added to the formerly mentioned variety of skills, programming styles and methods, but also to the cultural context and ethical dimensions of motivations to contribute to the open source development. However, in spite of this diversity, all the contributors "share the feeling to belong to a specific community motivated by a strong common identity", that counterbalance the weakness of their direct interactions and their dispersion (Ibidem). The cement of this cohesiveness is clearly the status of intellectual property that is a guarantee against deviations or misappropriation of the collective effort.

But the variety of the open source community members has also to be seen in terms of level of competences regarding the software development activity. First among developers, there are different degrees of involvement that varies from steady contributions in the core group to more occasional ones at the periphery. But an open source community doesn't restrict to the sole developers but also includes a category of "frontiers users" (Kogut and Metiu, 2001) that are not able to contribute to the software development but have an essential role in terms of source of innovations (Von Hippel, 1988) and above all testing and debugging base. "Hence, it is not modularity that gives open source a distinctive source of advantage, because it too relies on hierarchical development. Rather the source of its advantage lies in concurrence of development and debugging. In spite of its unglamorous nature, maintenance alone represents anywhere 50-80% of the software budget. The largest part of the developer community are not involved with code writing, but with code-debugging" (Kogut and Metiu, 2001).

Then it is clear that the status of intellectual property, as it is conceived in open source software, allows the development and working of a remarkably efficient production process in which all members benefit from the whole community efforts, while continuing to manage their proper activities on a decentralised way. Beyond the sole case of software development it is to be foreseen that such principles of working of "distant community" could have many applications in other knowledge intensive fields of activities. Diverse cases have emerged recently that could foreshadow a wider range of applications.

A first illustration is given by the "Wikipedia"[34] project of encyclopaedia. The project is born in 2001 in the US Internet company Bomis. It counts now for more than one million of entries, in a very large scope of languages : mainly English but also French (62 000 entries), German

[34] From Wiki meaning fast in Hawaiian and pedia from the Greek paideia meaning education.

(100 000), Japanese, … It is one of the most visited sites in the world. It is supported by gifts and voluntary contributions and a strict postcontrol permits to avoid content drift. It is the responsibility of a core group of "administrators" endowed with the trust from the whole set of the contributors. Telabotanica[35] project on botany is another example of collective production and sharing of knowledge.

An other significant illustration of this need to get out of a strict intellectual property rights protection is given by the recent iCommons initiative. It is an international movement born in the United-States, that is at the origin of the Creative Commons tool "to give authors free tools to enable them to mark their content with the freedom they intend their work to carry, while reserving the rights the author believes must be reserved.(…) It has attracted musicians, academics, authors, film-makers and researchers internationally who want a simpler way to exercise their rights without rejecting the protection of copyright altogether"(L.Lessig, 2004)[36]. Creative Commons Licences began to be translated and adapted to a wide range of national juridical contexts : Netherlands, Taiwan, Australia, Sweden, France … and have still been adopted by more than two millions of creative works in the world.

Levine (2004) emphasizes the spreading of collectives that "operate in concert to accomplish innovation goals that may have great economic significance, and accomplishment of those goals is often their primary *raison d'être*. The significance is clear : real products are created, services are offered, and economic rents are appropriated or lost". Such collective, that Von Hippel and Von Krogh (2002) call the "private collective" innovation model, can be mainly observed today as carrying their interactions on an Internet framework, but as put in evidence by Allen (1983) this type of "collective invention" doesn't require modern communication tools and can be observed in a more traditional context. In the British iron and steel industry (Cleveland district) in the second half of 19th century, companies have revealed freely among competitors technical information related to blast furnace design in order to allow an experimental incremental advance to improve the efficiency of industrial smokestacks. Osterloh and Rota (2004) also point out two other examples of "private collective" innovation models. With the Homebrew Computer Club formed at Stanford University in 1975 members exchanged ideas and projects in order to explore the potential applications of the already emerging microprocessor technology. Steve Wozniak, Steve Jobs and the other founders of Apple met in the club. An other example they give is the one of the early developments of flat panel display technology in the late 60s. Following Spencer (2003), a majority of the firms actives in this field have published at least one scientific paper then sharing a part of their RD results. These firms are precisely those who got the highest innovative performances (in terms of the value of patent portfolio).

Today, the biggest challenge for intellectual property rights mutualisation lies undoubtedly in the field of knowledge intensive activities with a high level of complexity of the knowledge base to be mastered and combinatorial aims of informations and skills. A significant example is the one of life sciences and industry. Joly and Hervieu (2003) plead for a high degree of knowledge resources mutualisation in the field of genomics in Europe, not for intellectual property renouncing, but by organising a collective system of management of intellectual property. This will permit to reinforce the competitive position of European firm facing US multinationals and, by pooling basic technologies, to avoid innovation clamping.

INSTITUTIONAL AND PUBLIC POLICY CONSEQUENCES.

The implications of these issues in institutional and public policy have to be understand at different levels. The main important one is un-

[35] http://www.tela-botanica.org/

[36] Lawrence Lessig is a Professor of Law at Stanford Law School. He also chairs th Creative Commons Project.

doubtedly the aspects of intellectual property status and protection. As noticed by the CONTU in the 70s it seams difficult to set up a new well fitted instrument of intellectual property protection. Even if other close domains like databases have been able to give rise to a *sui generis* framework, the main obstacle seems to lie in the necessity to reach an international consensus within a quite short time. The present debate appears rather to rest in the question of the patentability and the deeper institutionalisation of the "CopyLeft", GPL and other Open-Source licenses.

Intellectual property protection cannot be considered as an aim in itself and has to be treated in general interest considerations (CGP-Piéta, 2004). Presently the general trend around the world is the one of strengthening the protection as well in terms of duration as of scope. This doesn't work without any risk. History shows an alternation of strengthening and weakening phases of intellectual property juridical regimes. The present reinforcement especially visible in the accelerated patent granting, overall in the United States, reflects a relative falling of patentability criteria that "tends to distort competition and threaten to inhibit innovation. Should the occasion arise, this proliferation problem is particularly sharp for enterprises that less often play the role of depositor than the one of user of technologies designed by others. The induced cost has to be understood in terms of a substantial lost of time and juridical expenses" (Ididem).

As far as software patentability is concerned it seams important to conceive a clear position at national and European levels in order to clear up the too important ambiguities that derives from strong pressures of interest groups that aim to preserve their market dominating power. Three main aspects of the question have to be taken in account. The first one is the recurrent question of the impact of patents spectacular increase on innovation dynamics. It is the sense of intellectual property protection to foster innovation dynamics both by giving incentives to inventors and by improving the diffusion of ideas. It is a very sharp preoccupation to avoid that juridical tools that are mobilised for this aim wouldn't be at the origin of impediment to innovation. The second aspect is also related to the principles of the protection. It is important to note that software patentability, as it is applied, appears inconsistent with those foundations in so far as the source-code remains hidden, that doesn't respond to the obligation for the inventor to disclose the description of the technical aspects of the protected invention. Finally the third question is related to what should be and what shouldn't be protected. In other terms there is a large part of software components that are daily used by programmers and that should be considered as non appropriable in consideration of their character of pure knowledge[37] or of common resource. For this aim, a real state of the art should be necessary to determine the character of novelty of a patent claim.

It is clear that these questions are not dissimilar from those that can be raised in other domains, especially the one of life sciences and technologies, where the question of knowledge private appropriation raises problems of ethical, cultural nature and more simply of social efficiency. An unbridled patent granting on natural resources gives rise to the pillage of less developed countries biological resources by large multinationals firms[38]. The Netherlands, Italy and Norway have claimed in 1998 at the European Court of Justice against the European 98/4/CE directive on biotechnology patents on the motive of its infringement of the 1973 European convention on patents and the 1992 convention on biodiversity. This recourse has been rejected in June 2001 but the transposition of the directive in the French jurisdiction will introduce important restrictions by placing genomic sequences in the public domain and by insisting on the strict necessity of the industrial applicability criterion (Desbois, 2004). More recently, the European Office of Patents has invalidated in January 2005 the pat-

[37]The notion of "pure knowledge" has to be understood here as opposed to a patentable knowledge that requires an industrial applicability.

[38]The Rio de Janeiro Convention on bio-diversity in 1992 has pointed up the necessity to preserve the sovereignty of nations on their proper biological resources.

ent of the American firm Myriad Genetics on the BRCA1 gene of predisposition to breast cancer. With such a patent Myriad Genetics would have benefit from a monopoly position on detection tests at a price three times higher that the current price on European market. In addition it would have given to Myriad Genetics a power control on the related works driven by public laboratories in cancer research that would have need to be licensed and pay royalties for that.

In the field of genomics, the Human Genome Program, public international consortium launched in 1998 has sustained an attitude based on principles of publication and open diffusion of the results for the benefit of the whole scientific community. In France, the researchers demanding a support to the "Genethon" engage themselves to publish immediately their results rather than to take any patent. But this line has been progressively contested by the involvement of large private firms supported by huge financial resources (Orsi and Moatti, 2000). This can "lead to a de facto vertical integration of academic laboratories into the activities of these firms", like in the case of the significant agreement signed between one of the most prestigious laboratory of the HGP and the US firm Perkin Elmer for the creation of the joint venture Celera, chaired by Craig Venter. An asymmetrical competition engaged then between the public consortium HGP and Celera genomics, the latest using largely the data produced and published by HGP. In March 2000, US President Clinton and UK Prime Minister Blair declared jointly their position in favour of a free access to the whole raw data concerning the human genome. They had in mind to make pressure on Celera without excluding intellectual property protection for the future inventions based on genome knowledge. In Feb. 2001, HGP and Celera announced simultaneously their complete release of the human genome and published them respectively in "Nature" and "Science". From there, Celera has oriented his business activity on the use of its knowledge and skills in genomics for the development of new therapies and targeted medicine and has established, for that aim, a set of significant strategic alliances[39].

The other side of the institutional status of intellectual property is the question of the recognition and acceptability of the CopyLeft principles in the national and European juridical contexts. This includes the treatment of claims for infringement in the cases of abusive appropriation of open-source codes. As an illustration the French INRIA[40] with the CEA and the CNRS[41] have settled a new open-source licence called CeCILL -Ce(a)C(nrs)I(nria)L(ogiciel)L(ibre)- in order to offer an GPL-equivalent licence that could underlie contracts consistent with the French law. This initiative carried by public bodies has also a policy significance related to the feeling of interest for open-source software from those public bodies. Their commitment "can reassure some SMEs that would like to adopt those free software products but fear that such a choice could have pernicious effects on their own organization"[42]. While the official translation of the GPL is not yet achieved, the CeCILL license is available in French and in English and this conforms to the so-called "Loi Toubon" from 1994 Aug. 4th stipulating that contracts implying public bodies have to be written in French. Moreover, CeCILL specifies that for lack of conciliatory agreement, the potential lawsuits will be treated by Paris courts, what represents a consequent advantage for French developers and enterprises rather than having to take proceedings into a foreign court.

This dimension in favour of a real recognition of the CopyLeft could find expression in the official prerogatives of national, European and even international offices of intellectual property (like the French INPI -National Institute for Intellec-

[39] http://www.celera.com/celera/alliances

[40]National Institute for Research in Informatics and Automatics.

[41]Atomic Energy Department and National Centre for Scientific Research.

[42]Gérard Giraudon, chairman for industrial developments and relations at INRIA, in Y.Rocq "Faut-il adopter la licence CeCILL", Login, n°120, Sept. 2004.

tual Property- or the international WIPO -Word Intellectual Property Organization). They could assume competences in terms of information as well as juridical advice and assistance, as they do for standard intellectual property tools.

The second dimension of public policy concern is related to the adoption and use of open-source software in public administrations and services. In France, the prime minister has declared his intention to favour the choice for open-source software in the national administration[43]. In regions, some local governments, like the Regional Council PACA -Provence Alpes Côtes d'Azur- have announced their decision to progressively implement free-software in the high school establishments of the Region. Such a decision is not only a strong political signal in favour of free-software but also a mean to heighten the new generations' awareness of the principles of knowledge sharing and cooperative development. In the UK, the Office of Government Commerce announced in 2003 his intention to launch and coordinate a large "Proof of Concept" trials of Open Source Software in a range of public bodies in conjunction with IBM and Sun Microsystems. The final report pointed out the direct and indirect aspects of OSS advantages. "Apart from reductions in cost of software licenses, benefits of Open Source can include cost avoidance through reductions in the replacement cycles of hardware, improved software reliability and security, software platform stability, the ability to tailor and modify the software, easier administration, and greater scalability of hardware platforms" (OGC, 2004). Beyond the aspects of costs for public administrations and bodies, the possible turning towards open-source software adoption is thus also the mark of an involvement of the public sector in the process of open-source development, as a very large base for software products debugging and improvements, but also as a potential direct contribution to software development under GPL or equivalent licenses. As for IBM investment in Linux, these decisions boost the credibility of such offers and thus favour their diffusion.

Finally the third dimension of public policy is related to the actual support to open source software production. On the one hand this support can take the form of direct finance subsidies to free-software organizations or of placing developers paid on public funds to the disposal of open source projects. On the other hand there are a lot of computer codes that are developed in the context of the public administration or sector and that could be publicized under GPL or equivalent status without any problems of security or confidentiality. This would represent a contribution to the production of the open-source community and, at the same time avoid problems of duplication of parallel projects in the public sector, while benefiting from the possible improvements and testing of the open-source base. This idea is defended by the association Adullact which tries to organize the collective production and the exchange of the solutions produced between public communities in France[44].

CONCLUSION.

Software industry represents today a remarkable illustration of the fact that any intellectual property system is a compromise between different actors (the producer(s) of a good, their competitor(s), the users, the state, ….) A good balance between individual incentives to innovate and the maximization of the social utility generated by the diffusion and use of the innovation is very conditional to the technological environment and evolve in course of time.

Traditional intellectual protection framework allows producers to finance the initial phase of the innovation process, anticipating returns in the diffusion phase, in which they expect a legal monopoly to produce and commercialize it. Open Source movement shows that, for some knowledge intensive/cumulative innovation, this

[43]See on that topic the directive published by the French Ministry for infrastructure, transport, spatial planning, tourism and the sea : http://www.equipement.gouv.fr/bulletinofficiel/fiches/Bo200410/A0100067.htm

[44]http://www.adullact.org/

can be counter-productive, as the diffusion phase allow the producer to finance its initial investment without having a monopoly, but thanks to the feed backs and the joint needs generated by new users and uses.

This example, made possible at a global level by the existence of Internet has precedented occurrences in the history at more local scales. But with the diffusion of the Internet and the fact that research is more and more computer based, sharing knowledge and innovation can be much more efficient than even ten years ago. This could imply the need to stop the present trend of strengthening intellectual property protection regimes. And what happens now in software industry is going to spread to a large scope of other activities and industries.

Thus the question of the possible extension of such a model of production, based on knowledge openness and sharing represents a significant challenge for knowledge based industries in a near future. In the section 4 of this chapter, we have mentioned fields of knowledge intensive activities where the construction of a shared knowledge base is still a matter of fact, drawing advantage from the large diversity of skills and resources mobilized by the contributors for the benefit of every one of them and even beyond. So examples like Wikipedia (open encyclopedia http://www.wikipedia.org/) and Tela-Botanica (http://www.tela-botanica.org/) projects have proven that a collective production of knowledge helped by Internet is feasible in other contexts than software production. As we saw, this generates problems of free riding while opening the opportunity of a higher individual and social efficiency.

From there it appears possible to lay down two main rules that a model based on knowledge mutualism may conform, in order to aspire to viability.

- First the product of this knowledge collective production should not be the main purpose of sales but rather the ground on which a marketable activity could be built: cognitive input of a productive process, jointed services…
- Second, contribute to this public good production should give rise to direct rewards for contributors (like learning or a better fitting to personal needs in the case of OSS) and/or a competitive advantage on related markets.

When such conditions can be fulfilled, then public action can be of great deal for favouring the success of the model and should select the most appropriate ways according to the prevalent conditions of production and competition in the concerned industry: direct support to knowledge production (through financial aids and incentives, public contribution …), preference given to open source products on public markets, assistance to coordination, training programs, public information, stable juridical and knowledge protection environment…

Then the open source model could apply to a broader range of activities beyond the sole software sector, taking advantage from the huge potential of communication and cooperation drawn from Internet and the digitization of knowledge production. The key issue for this enlargement actually lies in moving the scene of value added production and competition, like it is the case for OSS. This would give rise to a renewed combination of actors' cooperation and competition ("coopetition") and a much higher social efficiency than in the monopoly model based on a strong intellectual property protection.

4.BIBLIOGRAPHIE.

[**R.Allen**, 1983, Collective invention, *Journal of Economic Behavior and Organization*, 4, p.1-24

A.Amin et P.Cohendet, 2004, *Architectures of Knowledge: Firms, Capabilities and Communities*, Oxford University Press.

W. B. Arthur, 1988. Self-reinforcing mechanisms in economics. In **P. W. Anderson**, **K. J. Arrow**, and **D. Pines**, editors, *The Economy as an Evolving Complex System*. SFI Studies in the Sciences of Complexity, Addison-Wesley Publishing Company, Redwood City C.A.

W. B. Arthur, 1989. Competing technologies, increasing returns and lock-in by historical events: The dynamics of allocations under increasing returns to scale. *Economic Journal*, 99: 116-131. URL: `http://www.santafe.edu/arthur/Papers/Pdf_files/EJ.pdf`.

C. Y. Baldwin and **K. B. Clark**, 2003. The architecture of cooperation: How code architecture mitigates free riding in the open source development model. Harvard Business School, 43 pages. URL: `http://opensource.mit.edu/papers/baldwinclark.pdf`.

J. Bessen, 2002. Open source software: Free provision of complex public goods. Research on Innovation. URL: `http://www.researchoninnovation.org/online.htm#oss`.

J.Bessen and K.Maskin, 2000, Sequential Innovations, Patents and Imitation, MIT, Department of Economics, *Working Papers* n°00-01, January.

CGP-Piéta, 2004, La propriété Intellectuelle dans le contexte de la société du savoir, *Le Quatre Pages*, N°1, 16 Mars, Le Plan, Commissariat Général du Plan, Paris.

W. M. Cohen and **D. A. Levinthal**, 1989. Innovation and learning: The two faces of r&d. *Economic Journal*, 99: 569-596.

L. Dahlander, 2004. Appropriating returns from open innovation processes: A multiple case study of small firms in open source software. School of Technology Management and Economics, Chalmers University of Technology, 24 pages. URL: `http://opensource.mit.edu/papers/dahlander.pdf`.

J. De Bandt, 1995. *Services aux entreprises: informations, produits, richesses*. Economica, Paris.

M. Delapierre, **L.-A. Gerard-Varet**, and **J.-B. Zimmermann**, December 1980. Choix publics et normalisation des réseaux informatiques. Technical report, Rapport BNI.

D. Demazière, F. Horn, and N. Jullien, 2004. Le travail des développeurs de logiciels libres. La mobilisation dans des « communautés distantes ». Conf. Proximity, Networks and Co-ordination, Marseille, June 17th-18th. To be published in Cahiers Lillois d'Économie et de Sociologie.

Desbois D., 2004, Vers une appropriation privative du vivant, Cadres CFDT, n°409.

G. Dréan, *1996. L'industrie informatique, structure, économie, perspectives. Masson, Paris.*

J.Farrell, 1989, Standardization and intellectual property, *Jurimetrics Journal*, Fall

J.Farrell et G.Saloner, 1988, Coordination through Committees and Markets, *Rand Journal of Economics*, Summer, Vol.19 N°2 : 235.

D. Foray, and **J.-B. Zimmermann**, 2001, L'économie du logiciel libre : organisation coopérative et incitation à l'innovation. *Revue économique*, vol. 51, octobre, pp. 77-93.

J. Gadray, 1998. La caractérisation des biens et des services, d'adam smith à peter hill: une approche alternative. Technical report, IFRESI, Lille. document de travail.

C. Genthon, 1995. *Croissance et crise de l'industrie informatique mondiale*. Syros, Paris.

C. Genthon, 2001. Le libre et l'industrie des services et logiciels informatique. RNTL. URL: `http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/workshop1/genthon.pdf`, workshop.

L.-A. Gérard-Varet and **J.-B. Zimmermann**, May 1985. Concept de produit informatique et comportement des agents de l'industrie. In *colloque «Structures économiques et économétrie»*, May 1985.

F. Horn, 2002. Company strategies for the freeing of a software source code: opportunities and difficulties. In **Jullien et al., [2002]**, editor, *New Economic Models, New Software Industry Economy*, pages 99-122.

F. Horn, 2004. *L'économie des logiciels*. repères, la Découverte.

N. Jullien, 1999. Linux: la convergence du monde Unix et du monde PC. *Terminal*, 80/81: 43-70. Special Issue, *Le logiciel libre*.

N. Jullien, November 2001. *Impact du logiciel libre sur l'industrie informatique*. PhD, Université de Bretagne Occidentale / ENST Bretagne, mention: sciences économiques, 307 pages. URL: `http://www-eco.enst-bretagne.fr/Etudes_projets/RNTL/documents_universitaires.html`.

N. Jullien, 2003. Le marché francophone du logiciel libre. *Systèmes d'Information et Management*, 8 (1): 77-100.

N. Jullien, **M. Clément-Fontaine**, and **J.-M. Dalle**, 2002. New economic models, new software industry economy. Technical report, RNTL (French National Network for Software Technologies) project, 202 pages. URL: `http://www-eco.enst-bretagne.fr/Etudes_ projets/RNTL/`.

P.B.Joly and B.Hervieu, 2003, La marchandisation du vivant, *Futuribles*, n°292, December.

B.Kogut and A.Metiu, 2001, Open Source Software development and Distributed Innovation, *Reginald H.Jones Center Working Paper* #01-08, April.

K. Lakhani and **E. von Hippel**, 2003. How open source software works: Free user to user assistance. *Research Policy*, 32: 923-943. URL: `http://opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf`.

J. Lerner and J. Tirole, 2002, Some Simple Economics of Open Source, *Journal of Industrial Economics*, 52 (June 2002) 197-234, http://www.people.hbs.edu/jlerner/simple.pdf

L.Lessig, 2003, Foreword, in D.Bourcier et M. Dulong de Rosnay (Eds.), *International Commons at the Digital Age*, Romillat, Coll. "Droit et Technologies", Paris.

S.Levine, 2004, Towards a Grounded Theory of Collective Open Source Innovation, Druid Summer Conference on *Industrial Dynamics, Innovation and Development*, Elsinore, Denmark, June 14-16.

A. Mocus, R. Fielding, J. Herbsleb, 2000, A case study of Open source software development: the Apache server. Proc. 2000 Int'l Conference of Software Engineering (ICSE2000), Limerick, Ireland, June 4-11, 263-272.

D. C. Mowery, editor, 1996. *The International Computer Software Industry, A comparative Study of Industry Evolution and Structure*. Oxford University Press.

L. Muselli, 2002. Licenses: strategic tools for software publishers? In **Jullien et al., [2002]**, editor, *New Economic Models, New Software Industry Economy*, pages 129-145.

O.G.C., 2004, Open Source Software Trials in Government, Final Report, Office of Government Commerce, http://www.ogc.gov.uk/embedded_object.asp?docid=1002367 .

F.Orsi and J.P.Moatti, 2000, Les relations recherché publique / industrie génomique : américanisation ou voie européenne?, *Médecine/Sciences* 14 : 94-100.

M.Osterloh and S.Rota, 2004, Open Source Software Development – just another case of collective invention?, SSRN Working Papers, http://ssrn.com/abstract=561744

O.T.A., 1992, «Finding a Balance; Computer software, intellectual property and the challenge for technological change», *OTA-TCT*-527, Whashington DC

J. Ousterhout, April 1999. Free software needs profit. *Communications of the ACM*, 42 (4): 44-45.

E. S. Raymond, 1998a. The Cathedral and the Bazaar. URL: `http://www.tuxedo.org/~esr/writings/cathedral-bazaar/`.

E. S. Raymond, 1998b. Homesteading the Noosphere. URL: `http://www.tuxedo.org/~esr/writings/homesteading/`.

E. S. Raymond, 1999. *The Cathedral & the Bazaar; Musing on Linux and Open Source by Accidental Revolutionary*. O'Reilly, Sebastopol, Calif.

G. B. Richardson, April 1997. Economic analysis, public policy and the software industry. In *The Economics of Imperfect Knowledge - Collected papers of G.B. Richardson*, volume 97-4. Edward Elgar, DRUID Working Paper.

P. Romer, 1993. The economics of new ideas and new goods. Annual Conference on Development Economics, 1992, Banque Mondiale, Banque Mondiale, Washington D. C., 1993.

G.Saloner, 1990, «Economic issues in computer interface standardization», *Economics of Innovation and New Technologies*, Vol.1, pp.135-156.

R. M. Stallman, 1998, The GNU project. *Open Sources*, O'Reilly ed.

J.W. Spencer, 2003, Firms' knowledge-sharing strategies in the global innovation system: empirical evidence from the flat panel display industry, *Strategic Management Journal*, vol.24 (3), 217-233.

C. Tzu-Ying and **L. Jen-Fang**, 2004. A comparative study of online user communities involvement in product innovation and development. National Cheng Chi University of Technology and Innovation Management, Taiwan, 29 pages. URL: `http://opensource.mit.edu/papers/chanlee.pdf`.

Vivant M., 1993, Une épreuve de vérité pour les droits de propriété intellectuelle : le développement de l'informatique, in *L'avenir de la propriété intellectuelle*, Librairies Techniques, Collection « Le Droit des Affaires », Paris.

E. Von Hippel, 1988, *The Sources of Innovation*, Oxford University Press.

E. Von Hippel, 2001, Learning From Open-Source Software, MIT *Sloan Management Review*, Summer.

E.Von Hippel and J.Von Krogh, 2003, Open source software and the 'private-collective' innovation model: issues for organization science, *Organization Science*, vol.14, 2

J.-B. Zimmermann, September 1995. Le concept de grappes technologiques. Un cadre formel. *Revue économique*, 46 (5): 1263-1295.

J.-B. Zimmermann, 1998. Un régime de droit d'auteur: la propriété intellectuelle du logiciel. *Réseaux*, 88-89: 91-106.

J.-B. Zimmermann, 1999. Logiciel et propriété intellectuelle: du copyright au copyleft. *Terminal*, 80/81: 95-116. Special Issue, *Le logiciel libre*.

LES BULLETINS RÉCENTS.

Année 2005.

8-2005. Jullien N., Zimmermann J.-B. New approaches to intellectual property: from open software to knowledge based industrial activities.

7-2005. Le Goff-Pronost M., Nassiri N. Deux approches nouvelles pour l'évaluation de la télémédecine : l'évaluation contingente et l'analyse multicritère.

6-2005. Trédan O. Les Weblogs dans la cité: entre quête de l'entre-soi et affirmation identitaire.

5-2005. Suire R. Encastrement social et usages d'Internet: le cas du commerce et de l'administration électronique.

4-2005. Thierry D., Trédan O. Cyberespace et affirmation des identités territoriales

3-2005. Guéguen N., Pichot N., Le Dreff G., Similarity and Helping Behavior on the Web: The Impact of the Convergence of Surnames Between a Solicitor and a Subject in a Request Made by E-Mail. Publié dans le Journal of Applied Social Psychology, n°35(2), 423-429.

2-2005. Farajallah M., LeGuel F., Penard T. Union Européenne élargie et nouveau voisinage : de la fracture numérique à la coopération numérique ?

1-2005. Granjon F. Champ d'Internet, pratiques télématiques et classes populaires.

Année 2004.

1-2004. Cardon P., Trellu H. Les personnes vieillissantes et la télé-assistance: privilégier la dimension relationnel.

Responsables de l'édition: Godefroy Dang Nguyen, Nicolas Jullien.

Contact : Nicolas Jullien

M@rsouin
GET - ENST Bretagne
CS 83818, 29238 Brest CEDEX 3

Marsouin@infini.fr
(0)229 001 245