

Free/Libre/Open Source Software (Floss): Lessons for Intellectual Property Rights Management in a Knowledge-Based Economy

Nicolas Jullien, Jean-Benoît Zimmermann

▶ **To cite this version:**

Nicolas Jullien, Jean-Benoît Zimmermann. Free/Libre/Open Source Software (Floss): Lessons for Intellectual Property Rights Management in a Knowledge-Based Economy. ICFAI journal of cyber law, 2007, 6 (3), pp.19 - 36. hal-01213698

HAL Id: hal-01213698
https://hal.science/hal-01213698

Submitted on 8 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Free/Libre/Open Source Software (FLOSS):

lessons for intellectual property rights management

in a knowledge-based economy.

Nicolas Jullien – M@RSOUIN
Jean-Benoît Zimmermann – CNRS / GREQAM et IDEP

This version : November, 2006

Paper prepared for the DIME London Conference
Intellectual Property Rights for Business and Society
14th and 15th September 2006

Abstract : The aim of this paper is to focus on the emerging situation in which open source software is nowadays produced not only by individual developers but in a growing proportion by firms that hire programmers for their own objectives of development in open source or for contributing to open source projects in the context of dedicated communities. As commercial firms it is important to analyze how and why they are capable of drawing benefits from such involvement and their connected activities. Moreover, we want to stress the different types of business model these firms rely on and the possible evolution they are likely to follow in the near future. We shown how Open Source principles provide an alternative way of thinking and managing intellectual property that do not come up against the same problems but needs a radical change in the way of drawing commercial benefits from knowledge development tasks. Then we analyze the growing involvement of commercial actors by setting up a typology of the different business models that can be observed in the OS landscape, how they correspond to the different strategies of industrial firms according to the main characteristics of their technical skills and market position. Finally, in a conclusive section we will draw the main lessons of the FLOSS experience for a possible enlargement of those principles of IPR management and business to other knowledge based commercial activities.

1. Introduction

A “free”/“libre” or “open source” software (FLOSS) is a software whose source-code, that is the explicit expression of the programming work, remains openly accessible. It appears as an alternative solution to the question of intellectual property in the computer software field, in which neither copyright nor patents can bring an acceptable balance between innovation incentives and knowledge diffusion. Since Copyright protects not the ideas but a given expression of the ideas, software editors do not generally reveal the explicit expression of the programs (the source code) and sue those who try to disclose it. This behavior impedes knowledge diffusion in contradiction with the principles of IPR protection and handicaps the accumulative characteristic of innovation and software products interoperability. On the other hand patenting software could lead to a progressive partitioning of the field into proprietary owned procedures or algorithm and contradicts the recent evolution of programming techniques that are based on a closer relation to scientific knowledge and a combinatorial assembly of reusable components.

On the contrary, the alternative model of Open Source Software -OSS-, is based on a very innovative juridical concept called GPL "General Public License" and its diverse variations, and consists in forcing the producers to disclose both the source-code of the concerned programs and any further improvement if they are re-distributed/re-sold. It corresponds to a totally different approach of intellectual property rights, based on a weaker protection and the ability for all the actors to benefit from the whole set of innovations and progresses from a shared knowledge base. Of course, as in any public good question, this immediately raises problems of possible free-riding and of the incentive to disclose such knowledge in so far as accessing to knowledge doesn't depend on having contributed or not. That's the reason why the viability of this new form of IPR management will depend on the sustainability of associated business models and institutional supports.

Until recently, FLOSS was considered as only concerning programmers motivated by the building and the sharing of a base of programs developed for their own needs. Today, the open source model involves commercial enterprises and also an enlarged market of simple users. This brings us to a paradoxical situation in which the development of business relies on the existence and durability of an activity of non-market nature. In former works, we have shown that solving such a paradox requires the setting up of new modes of incentives involving a pecuniary dimension additionally to the motivations of programmers originating in the initial movement. Such a turn still appears to be part of the current way of working of FLOSS in so far as a growing amount of the code is produced by employees who are paid for doing so. Such “hybridization”, mixing market and non-market rationales, nowadays appears as an inescapable evolution that also challenges policy makers to integrate support to FLOSS in the instruments of technological policy. It is then of growing importance to better understand under which conditions such a model of IPR management could extend to a growing number of knowledge intensive economic activities.

The aim of this paper is to focus on the emerging situation in which FLOSS is nowadays produced not only by individual developers but in a growing proportion by firms that hire programmers for their own objectives of development in open source or for contributing to open source projects in the context of dedicated communities. As commercial firms it is important to analyze how and why they are capable of drawing benefits from such involvement and their connected activities. Moreover we want to stress the different types of business model these firms are relying on and the possible evolution they are likely to experience in the near future.

Section 2 is devoted to the analysis of IPR traditional forms of protection in the software industry and their failure. We will explain how Open Source principles provide an alternative way of thinking and managing intellectual property that do not come up against the same problems but needs a radical change in the way of drawing commercial benefits from knowledge development tasks. In section 3 we will describe how FLOSS has progressively switched from a contribution model based on individuals' benevolent efforts to an actual industrial one. Then section 4 will aim to set up a typology of the different business models that can be observed in the OS landscape, how they correspond to different strategies of industrial firms according to the main characteristics of their technical skills and market position. Finally, in a conclusive section we will draw the main lessons of the FLOSS experience for a possible enlargement of those principles of IPR management and business to other knowledge based commercial activities.

2. The failure of the standard IPR protection means and the FLOSS alternative.

The question of intellectual property rights protection for computer software was raised as soon as software products could, in the mid 1970s, be considered as commercial goods in their own right and not only as application technologies linked to the market for computer systems. Following the United States in this regard, Europe and Japan adopted various frameworks of copyright laws which differed according to national legal context as well as differing cultural attitudes towards intellectual property.

But software IPR protection is still not satisfactorily settled by the copyright protection owing to the very specific nature of the software product and its production conditions. First of all a software product can be considered as an intellectual expression of ideas that are coded by the use of a specific programming language, with its proper vocabulary, syntax and structural rules. For this reason its protection has been considered as falling in the field of copyright. From a practical point of view however, a software program aims at carrying out a given task relying on the resources of the computer it is implemented in. Alternatively, in the case of a software system, the aim is 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 no longer readable. If the source-code is not supplied jointly it can only be restored imperfectly and at some significant cost, through a heavy operation of reverse engineering. Implemented on a machine, the program is able to properly emulate 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 reach an acceptable compromise between granting incentives to the inventor through temporary monopoly rights on the commercial exploitation of his invention and favoring 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 of the ideas and not the ideas themselves. Software producers are therefore not obliged to disclose the source-code of the protected programs. Most of the editors commercialize 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 constrained at all to reveal any information on the working principles of the protected program. US jurisprudence has adopted a quite severe attitude in this regard, strengthening the protection, by condemning for copyright infringement any suspected attempt of reverse engineering on copyrighted programs. So the use of copyright to protect software creates a distortion, since companies can enjoy protection while keeping secret the object of this protection. "For the first time since Sybaris, 500 B.C., who imposed public disclosure in exchange for the legal protection of recipes, private property and secret are reconciled!" (Vivant, 1993). The problem here is to extend copyright protection to an object which is fundamentally different from artistic and literary works. The purpose of software is not to communicate the expression of the ideas and inspiration, but to command and control a machine.

In spite of the juridical preference for copyright law expressed in the seventies, an increasing number of patents for software programs or even simple procedures or algorithms were granted during the 1990s in the Unites States. In Europe, the European Office of Patents remained on its initial position to grant patents only when they are an integral component of an industrial device or process. However, the European Commission has more recently submitted a new Directive aiming at 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 of all along their development work has been considered until now as belonging to the public domain and freely available. In the absence of a

real state-of-the-art in the field of computer software programming, patents are granted in a totally arbitrary way, giving private rights for the use of resources that had been formerly shared by professionals 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 Maskin, 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 nowadays since increasingly 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 requires both an increasing recourse to a large scope of software components, portable and reusable in different contexts, and a growing proximity to the mathematical foundations of programming. This evolution makes the problem of the distinction between public and private property of modules and algorithms more acute. Copyright laws have rejected principles and algorithms from the scope of protection. Patent granting for software components creates, however, a barrier to their usage and contradicts the working mode of the whole community of software developers. The sole actors who will be able to manage such a situation are the large companies that will have the capacity to build a large portfolio of patents. The cost of the defense will be so high for SMEs that an attack for patent infringement may threaten their survival[1]. 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 the firms that are newcomers in the information technology field and that have understood the opportunity to build the basic material for future speculative profits at low cost[2]. It is also a real threat for open source software as illustrated by the SCO case. The 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 claimed one billion dollars from IBM and sent a letter to 1500 other companies to inform them of the risk they run when they continue to offer solutions derived from GNU/Linux[3].

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. The basic distinction with traditional industrial activities is that now the main input of the production process is of an informational and cognitive nature.

This opposition between private property of the codes, that gave rise to the software industry, and the free circulation of the sources considered as pure knowledge, that corresponds to the tradition of "open science", had already proved to be divisive in the software developers community like at MIT in the mid-1970s (Smets and Faucon, 1999). It is at the origin of the birth of a "free software" movement at the beginning of the 1980s, that aimed to preserve the diffusion of ideas and the combinatorial and cumulative nature of technical progress, both in terms of concepts and tools and in terms of algorithms for problem resolution and methods of coding.

From that situation stemmed the definition of an "open-source" software product as a program whose source-code has to be freely accessible and cannot be privately appropriated. In that sense open-source software fits the definition of public good insofar as it is a non-rival and non-exclusive product. With the birth of the "Free Software Foundation" and the launching of the GNU[4] project, by Richard Stallman in 1984, the first collective development project had as its aim an open Unix-equivalent platform. It appeared necessary to build up the

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

[2] It is the case of the US company Acacia, formerly start-up incubator whose sole activity is 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

[3] 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

[4] GNU's Not Unix

legal framework that could guarantee these principles of "CopyLeft" based intellectual property of software. Next, the GPL or "GNU-General Public License" was designed in order to protect the foundations of cooperative work development and to prevent any private appropriation of part or all of the concerned code lists, as may be done with software which can occur in the public domain. Hence, through the GPL, intellectual property is not rejected, authors do not renounce their rights but just the monopoly rent, which such rights would produce 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 authorize anyone who wants to make use of their work (modifications, improvements, additional features...) to do so under the sole condition that the new product must also circulate freely.

Of course, GPL has a seminal role in opening new principles of intellectual property management. But if it doesn't necessarily fit to the needs of any actors of this emerging open source world, particularly to those of the commercial firms that decide for different reasons to join the Open Source alternative. Many "hybrid" licenses have been designed in order to reconcile cooperative development and private interests in a variety of specific contexts. They involve different ways of combining the copyright and copyleft rules in different proportions (Smets and Faucon, 1999, Muselli 2002).

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 some doubt can be cast on the efficiency of these open but not completely free licenses, since users may not trust the producer is really willing to keep the sources open and to cooperate[5].

The other institutional side of the status of open 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[6] with the CEA and the CNRS[7] settled a new open-source license called CeCILL -Ce(a)C(nrs)I(nria)L(ogiciel)L(ibre)- in order to offer an GPL-equivalent license that could underlie contracts consistent with French law. This initiative carried out 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"[8]. While the official translation of the GPL has not yet been achieved, the CeCILL license is available in French and in English and this conforms to the so-called "Loi Toubon" of 4 August 1994 that stipulates 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. This represents an important advantage for French developers and companies which do not have to take proceedings into a foreign court, even if it may hinder foreign collaborations[9]. At the European level, a recent report to the European Commission stresses that GPL and other OSS license do not meet the requirements of the European Union legal framework and therefore there is a need to set up a well-fitted license. "The GPL's major problem is that the right of communication to the public is not provided explicitly

[5] Soufron and Sallantin (2005) analyse the increasing variety of free/open source licenses as different combinations of the four types of requirements : 1. the right to access (to the source code), 2. the right to modify, 3. the right to redistribute and 4. the right to use.

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

[7]The CEA is the French acronym for Atomic Energy Department and the CNRS if the French National Centre for Scientific Research.

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

[9] As they do not know the license, they will not be sure that their work would not be appropriate by the French producer.

amongst the granted rights, and that a clause limits furthermore the granted rights to what is explicitly provided by the license. Moreover, the GPL is known for being the most viral license ever, whereas massive spreading through dynamic linkage is not the aim of the European Commission." (IDA/GPOSS, 2004, p.3)

3. From an individual's benevolent contribution to an industrial based model.

As far as FLOSS production was limited to the audience of a community of developers apart from the commercial sphere, it only met a very small part of the users' needs and had no actual economic significance. Things however turned to a radically different situation with the appearance of new adopters, a new demand from simple users who aimed to benefit from the development efforts of the FLOSS community without being in debt for any counterpart, at least at a monetary level. Such enlargement of the base of users concerned was due to the conjunction of three complementary factors. First it is related to the onset of FLOSS mature products with a high level of performance and reliability. Secondly this enlargement was facilitated by the strength of diffusion inherent in Internet and that acts additionally to the interconnection capacity of the developers' community. Last but not least the arrival of commercial enterprises devoted to the distribution of FLOSS edited with more sophisticated designs, user interfaces and manuals, tutorials etc, is of first importance for a category of users with a weak level of technical culture and not so used to surfing on Internet. This new category of actors in the FLOSS world found its own way to reconcile commercial requirements with the principles of non-appropriation by basing its profitability on selling related services like users' on-line support, updating, debugging, information systems engineering, training...

What appears important here is to understand how this audience enlargement has impacted the developers' motivations and the way it can alter the working of the FLOSS community. It is clear that consequences can be identified as two contradictory effects.

First it can be considered for the developers as a satisfactory sign in so far as it is a result of the successful achievement of the objectives of the FLOSS community. The level of adoption of FLOSS products is a consequence of their intrinsic qualities that incites some of the users to switch from a former proprietary market (or to choose a "FLOSS" product rather than a proprietary one). As a consequence, this tends to reinforce the weight of the open source option, hence the conditions for its practicability in its competition with the proprietary model. Most of the FLOSS contributors are, either explicitly or not, cut-throat opponents to the Microsoft dominant position and are not reluctant to gain converts. These new users take part ipso facto in the achievement of a critical mass likely to reinforce the future competitive strength of FLOSS. Another positive aspect of the users' population enlargement is that it provides a wider testing and improvement base, the sole question being that of gathering the relevant information and transmitting it to the developers' community[10].

Conversely, other aspects can play a negative role. Of course one may think of the free-riding attitude these new users could be accused of having by drawing benefit from public goods without having contributed to its production. But this is not a real problem because most of these simple users wouldn't have been able to contribute anyway and this is amply balanced by the positive effects of their presence that have been evoked and it can be considered of no impact on the incentives structure of the developers. But this new demand is at the origin of a new business potential about which it is to be feared that free riding behavior of opportunistic firms could take shape by drawing private benefits from the marketing of products they didn't help develop.

Of course the developers' motivations are not only of a utilitarian nature. Lakhani and Wolf (2003) have completed a study from a base of 684 programmers involved in 287 projects and have shown that intrinsic motivations like the taste for creativity or intellectual stimulation play a major role for most of the developers. Nevertheless, this new situation has led to the confrontation of two worlds based on so opposed working rules that it is bound to be of consequence. Voluntary and not expecting too much in terms of revenue, FLOSS developers then had to interact more and more frequently with entrepreneurs with much higher income

[10] See Hapke, Jullien and Zimmermann (2005)

perspectives. Sometimes the developer becomes entrepreneur, sometimes he is only the supplier of the products that will give rise to market activity. Almost always appears a gap that actually affects the tempo and style of life but also the level of revenues. This undeniable fact is likely to call into question the structure of incentives to contribute for the developers by giving a renewed importance to utilitarian considerations. It can then lower the developers' mobilization hence the efficiency of the cooperative process. As a consequence, frustrated by this confrontation, some of the developers may leave of the game (Foray and Zimmermann, 2001) or allocate a growing part of their time to more lucrative activities related or not to the FLOSS world. To better understand this question we should look carefully at the more economically driven dimensions of contribution incentives.

It has to be recalled that, at the origin, the main motivation for developers was to have at their disposal the products they needed and to avoid duplicating efforts of software development or improvement. In other words and more widely, the collective effort of development generates a usage externality that anybody within the community of developers can capture. Thus it can be classed as a public good, non-rival and non-exclusive[11]. At a collective level it is a motivation for cooperating but at the individual level it can motivate defection and free-riding. Capturing the externality does not depend on having contributed or not; so, like in a standard model of public good production, individuals only contribute when the marginal utility of their contribution is greater than its marginal cost. Production function being usually of concave shape, this implies a limited total amount of the individual efforts. Nevertheless the marginal cost of the cooperation remains often very low. Lakhani and Von Hippel (2003) show this from an on line system of inter-users assistance. Thanks to the growing storage capacities and transmission potential given by Internet, it works at insignificant costs. If the system is large enough, there is almost always a user that can provide at a minimal cost a solution to a problem raised by another user (the probability of a solution existing somewhere in the system is high) and the transmission cost is almost nil.

But additionally to the usage externalities and ethical motivations two other kinds of individual incentives have to be taken in account: learning and reputation.

In terms of learning, 42% of the programmers questioned by Lakhani and Wolf (2003) consider the improvement of their individual skills as a significant reason to contribute, which ranks this "extrinsic" feature in second place of developers' motivations. Programming is a non-stabilized art whose methods and procedures remain often faintly codified and of large diversity, even within the productive organizations of large software editors[12]. Following and taking part in the efforts of a FLOSS community bring feedback effects to the programmer in terms of improvement of his programming skills for at least two reasons. The first stems from a learning-by-doing effect, drawn for the involvement in collective effort of development. The second is due to the confrontation with an evolving code issued from a diversity of contributions related to a wide range of skills, methods and styles of programming. Such learning by interacting is a source of progress in the programmer's abilities, and progress greater than any made from his involvement in a more traditional development team, even within large companies where software production organization remains compartmentalized and divided in small teams. This learning by interacting is very significant for a programmer wishing to withstand the challenge of the knowledge evolution and this all the more because it takes place in the context of a repeated game.

Reputation effects play a complementary role to the former[13] and work in a similar way as in the academic sphere. This stems from the recognition by peers, in so far as accepted and labeled changes or improvements in an open source code are signed by their author, as explicitly expressed in the circulating copies. By contrast with the academic world, the main attribute of this recognition capital is to be converted in pecuniary terms from possible job with a firm or a better access to a funding source.

[11] This non-exclusive characteristic is effective within the population of developers. Beyond this population it is related to the technical skills of the user. So simple users can need specific tools, interfaces, support or training that give rise to a new demand on the market.

[12] Zimmermann (1998).

[13] Lerner et Tirole (2002) gather in a same category called "signal incentives", two types of effects "distinct though difficult to distinguish", on the one hand the reputation effects and, on the other hand, the satisfaction effects ("ego gratification") directly issued from the peers recognition.

Foray, Thoron and Zimmermann (2006) show that incentives related to learning aspects work differently depending on the level of competences of the individual developers and in complementarity to the benefits expected from a commercial exploitation of the software products. Typically, a less experienced individual will be more attracted by his own skills improvement, while an accomplished programmer will link his contribution to future earnings related to the commercial services related to FLOSS products diffusion. In a vertical differentiation model à la Shaked and Sutton with heterogeneous consumers (from their consent to pay for those services quality), the latter will depend on the quality of the available FLOSS product and the level of competence of the service provider. Then the most skilled developer contributes more, sets a higher price and gets a higher payoff. So, market perspectives do not lead necessarily to opportunistic behaviors but can play, for the most skilled, an inciting role in complementarity to the learning expectations for the less experienced developers[14]. As Demazière & al. (2005) have shown, these two roles are not opposite, but rather represent two steps, two periods in a developer's life (or "career").

Of course, it would be rather naïve to think that a pronounced taste for entrepreneurship from practiced developers would naturally originate from this result. Reality is always more subtle and combines individual and enterprise rationales. Von Hippel (2002) reveals that the members of the FLOSS community are little inclined to invest time and money in an entrepreneurial venture. This latter type of incentive should probably give rise to enterprise creation for a very limited proportion of developers while a large proportion of these highly skilled developers will be hired by firms that would find an economic interest to an involvement in both market and non-market spheres. Those developers will continue to contribute to the Open Source efforts for part or all of their wage-earning time, while benefiting from earning levels in concordance with the industrial rates at their actual qualification level. So Lakhani and Wolf (2003) notice that "a majority of our respondents are skilled and experienced professionals working in IT-related jobs, with approximately 40 percent being paid to participate in the FLOSS project."

So company involvement in the FLOSS world appears increasingly as a natural substitute to the individuals' voluntary contributions based model. This is all the more important since these new FLOSS industrial actors are endowed with a key responsibility for gathering and analysing the huge amount of information originated in simple user FLOSS product utilisation and transferring it to the developers' communities[15].

4. From business models to industry structure.

The central question for economics turns to understanding why firms deliberately get involved in FLOSS production thus abandoning any monopoly power of the fruit of their efforts, then any perspective of durable profits by selling those products as such, but on the contrary allowing a free access to them for users and potentialcompetitors. Our aim in this section is then to better understand how firms in each segment in the IT industry can cope with this new way of managing its IPR, and the possible impact of their strategic position on the global industrial structure.

But to do this, we have to take in account the recent evolution of the IT industry. During the 1990s, with the arrival of the Internet, the main technical evolution in information technology was, of course, the generalization of computer networking, both inside and outside organizations. Miniaturization also allowed the appearance of a new range of "nomad" products like Personal Digital Assistants (PDA, such as Psion and Palm), mobile phones, music players,... that have change the modes of using IT products and their interrelatedness within daily life. First of all, network communications and exchanges between heterogeneous products and systems are nowadays crucial and require appropriate standards. To this aim open solutions are probably the best guarantee for users and producers for product reliability throughout time and the successive releases of the products. A second aspect stems from the wide diversity of users and users' needs that require software programs (and more particularly, software packages) to be adapted to the needs and skills of every individual without losing the economies of scale. What characterizes the technological evolution of software is

[14] "The differences between the two groups (of contributors) are consistent with the roles and requirements of the two types of F/OSS participants. Paid contributors are strongly motivated by work-related user need (56%) and value professional status (22.8%) more than volunteers. On the other hand, volunteers are more likely to participate because they are trying to improve their skills (45.8%) or need the software for non-work purposes (37%)" (Lakhani and Wolf, 2004).

[15] See Jullien and Zimmermann (2006b).

thus the increasing interdependence between software programs built from basic components and modules that have to be more and more reused thus becoming increasingly refined and specialized (Zimmermann, 1998). Furthermore the related demand for software customization generates a renewed services activity for the adaptation of standard-component software programs[16].

There is a large diversity of actors in the industry as well in terms of products as in terms of size. Lots of innovations and company strategies have led to a progressive reshaping of the industry borders and structure. However, the foundations of the industry remain unchanged, as stated by Gérard-Varet & Zimmermann [1985] and Zimmermann [1995]: computers are built by assembling hardware and software units in a given architecture, and these computers (isolated or integrated into networks) are used as parts of information systems and solutions. On the base of such technical organization, it is then possible to distinquish four main types of "vertical specialization": component producers, computer and IT devices suppliers, software editors, and services companies providing customized software and integration.

So we will organize the discussion by making the distinction among those four large categories of actors. For each of them we will first clarify the basic conditions of each type of activities, characteristics of the products and characteristics of the users and hence of the demand. This will enable us to draw the main aspects of the market structure: base of added value, competitive advantages and nature of the competition. This study will help to understand the actors' IPR management strategies, especially regarding open source IPR management adoption, and the incentives to use and to contribute to the development of FLOSS products. We will then examine the consequences for future of FLOSS development and its industrial durability.

4.1. Hardware components producers.

They supply the basic components of the electronic devices like chips (Intel, Toshiba), hard disks or cards (ATI: graphic cards)... This hardware components industry is characterized by substantial economies of scale due to the importance of fixed and sunk costs (R&D, plants...) In each segment, competition is based on innovation (products performances and features) and the performance / price ratio. In addition a growing number of users pay attention to the main components that are incorporated in the electronic device they buy as a critical feature of the performance of the product. So the brand reputation is also a decisive competitive feature that reinforces the oligopolistic nature of the industry for every component sub-market.

In a first approach, these firms could be considered as weakly concerned by IPR management on software. But they develop 'drivers' for their own product to give them the ability to interoperate with other components and to be managed by the operating systems. So, their incentive to use and develop FLOS drivers for free operating systems (such as Linux) is a growing function of such systems market size. Since the beginning of the 2000s, some firms like ATI indeed offer such compatible drivers.

But, this remains a marginal contribution, and should not have any immediate serious impact on the structure of the FLOSS development organization.

4.2. Computers and IT devices suppliers.

Using these components, firms build machines, more or less dedicated to specific uses. At one extreme computers can be used for a wide scope of applications provided the software that is acquired and installed on them. At the other extreme, video game consoles or multimedia players are devoted to a single range of applications, while in between, mobile devices like PDAs or mobile phones are built to support a growing number of applications[17].

[16] The share for IBM service turnover jumped from 32 percent to nearly 53 percent in eight years (1997-2005, see http://www.ibm.com/annualreport/2005/). To explain that, IBM states in the same report that "Clients increasingly seek solutions rather than "point-product" purchases of particular technologies and products » (p. 6). HP tried to re-purchase the consulting part of PriceWaterhouseCoopers, which IBM finally acquired. Compaq, since re-purchased by HP, announced in June 2001 that it was giving itself 18 months to become a "service company".

[17] This distinction between specialized and generalist devices is evolving, as Sony intends its PS3 to be the media center at home. But this has not so far impacted on the industrial structure.

4.2.1. Servers.

These computers are intended to manage, deliver and protect information on the networks. They must be high-performance, stable, but also compatible with network standards. An archetype of server producers is SUN.

This segment of industry is characterized by notable economies of scale due to substantial sunk costs (R&D), but also by learning effects, and technological interrelatedness (the more programs running on the machine, the bigger the market potential). Firms sell hardware, but more and more usage characteristics (high-performance, reliability, "evolutivity"). The users are computer literate people, whom we have called "sophisticated users", or VH ie. "von Hippel users"[18], as they are able to express needs in technical terms, to develop software for their own needs, and to innovate by themselves.

Earnings in this market come from the computers sales, but also from services of maintenance and assistance to the user.

The competitive advantage comes from the installed base (thanks to technological interrelation inducing switching cost when a user wants to migrate to another provider) and the application portfolio, the distribution channel and the brand reputation, reflecting the hardware performance, the quality of the technical staff (to produce innovation and the right assistance to users). The industry structure is a stable oligopoly with important barriers to entry and appropriate role of quality.

But new strategies, based on more open architectures (PC servers and FLOS operating systems) have been emerging, since the beginning of the 2000s, when IBM, followed by HP announced it would implement Linux on its machines.

The incentives to use FLOSS are quite easy to understand:

- first the diffusion path. Traditional server manufacturers (such as IBM, or HP) are reproducing the same type of behavior that they had in the past vis-à-vis innovations such as Unix or microcomputers. Aware that there is a demand, they seek to integrate the new offer in their own offer portfolios, as they did with the preceding innovations. So doing, they legitimate the FLOSS offer by placing it at the same level as the other operating systems and thus facilitating its diffusion;
- doing so, they adopt a classical "challengers strategy" as they try to support a product which is in competition with the dominating standard they do not control. This can be understood as a reply to the success of Microsoft and its dominant position on the PC market, but also as a consequence of the arrival of new competitors like Dell selling PC servers and of the success of SUN in the Unix market[19];
- finally firms whose activity depends on a standard they do not control directly have an interest in choosing the most open as possible, so as to avoid dependence on the strategy of its owner and to stay aware and involved in its evolution.

This does not directly really impact the core market of "global players", firms like IBM, more and more oriented on service provision (maintenance, hotline) as we shall see below, but rather smaller or less diversified firms like SUN.

In that new scheme, incentives to contribute to FLOSS development are fourfold:

- this is the best way for a server manufacturer to ensure hardware-software compatibility, in a market where performances and sustainability are very important,
- it is the only way to ensure users that their needs will be integrated into the future versions of the FLOSS product,
- as explained by Cohen and Levinthal (1989), the participation in the development increases the absorptive capacities, and thus the comprehension of a knowledge-based good. In a market where quality insurance and on-time delivery matter, this is of importance,

[18] In reference, of course, to its work on "users as innovators" (von Hippel, 1988) and specifically on FLOSS production (Lakhani and von Hippel 2003, for instance).

[19] It worth noting that, on the contrary, SUN, being the leader on the UNIX market, has been reluctant to adopt Linux and is today the server constructor which has the most difficulties to adapt is business model, with recurrent losses.

- in a technical market, contributing is also a signal towards clients as reflecting the firm's technical skills.

This may have a strong influence on the FLOSS development organization. Today, companies are more and more involved in Linux, or Apache development, directly with the involvement of their own developers, or indirectly, by funding foundations[20]. Thus these development are increasingly done by a consortium of firms, rather than an open community of users.

Concerning the probable evolution of this server sector, the availability of free software programs for this market has a double impact. On the one hand, it contributes to generate a standard Unix offer independent of the platforms where it is implemented, and it reinforces the attractiveness of the Unix systems (in so far as Unix is the privileged support of free-use software programs, even if most of them also work under Windows). On the other hand, it puts more pressure on Unix computer manufacturers towards the unification of the Unix world around GNU/Linux that offers the additional advantage of running on PCs (Jullien 1999).

4.2.2. Microcomputers.

These machines (with a growing market share for laptop computers) are used by end users, mainly as personal computers. In order to clarify the analysis we will split this segment into the two categories of "*quality computers" -QC-* targeted at organizations or intensive end-users[21] and "*low price computers*" *-LPC-*, targeted at basic uses (Internet uses, for instance) and low skilled users. Market structures, actors, and IPR management are dissimilar. But some of the suppliers are selling on both markets by differentiating their product ranges in order to draw benefits from enlarged scale and scope economies. This appears quite similar to other industries like the automobile industry where top- and bottom-of-the-range markets do not work similarly but can be partly held by the same manufacturers.

An archetype of "quality computers" supplier could be Dell, or Toshiba, while an archetype of "low price computer" could be Acer or Packard Bell.

The microcomputer industry is characterized by significant economies of scale due to high fixed costs (automated plants), but also to the prices negotiated with component producers that have a high volume elasticity. Competition on the QC market is rather based on quality (performances, reliability, evolution, weight, battery life time), while it is based on prices in the LPC market. QC users are somewhat less computer literate than server users; we will call them "intensive frontier users[22]". Conversely, LPC is a mass market, where users have no particular skills.

Earnings in these markets come from the product sales.

The competitive advantage, additional to the capillarity of the distribution channel, comes from quality-price ratio for QC manufacturers, and from the price for LPCs. By the time manufacturers have become technology takers and follow the dominant standards as well in terms of hardware as of software components, such as MS Windows. Thus, they do not have actual IPR strategies regarding software.

However, incentives to use FLOSS could become far from negligible, for quality and differentiation purposes (easiness to develop maintenance and quality insurance services) on the QC market, for price purposes (no license fees for the operating system or the office suite) on LPC.

Thus, if in the price driven market incentives to contribute to FLOSS development do not exist, they may emerge for QC manufacturers, for similar reasons to those of the servers market, all the more that producers are often the same in both markets.

[20] For instance, Linus Torvald, creator and leader of Linux project, works for Open Source Development Labs (OSDL), an industrial consortium "dedicated to accelerating the growth and adoption of Linux in the enterprise" http://www.osdl.org/.

[21] People playing games, watching video films on their computer.

[22] Intensive for the intensiveness of use, and 'frontier' in reference to Kogut and Metiu (2001) definition.

However, as far as we know, there has not been yet any FLOSS based offer in the microcomputers markets coming from the main players[23], even if some suppliers propose Open Office or Mozilla to be run in a Windows environment on their machines.

4.2.3. Dedicated devices.

As already explained, information systems are nowadays characterized by networking of numerous kinds of machines, computers, but also dedicated machines such as PDAs, multimedia players, mobile phones, ... Most of these devices are connected to the computer(s), but more and more they also have access to the Internet (via Wifi, local networks or adsl technologies), and share applications with the computer (email, personal data management, etc.) Even devices like video game consoles are increasingly connected to an Internet-based network.

Archetypes of this evolution are Nokia's mobile phone, Nintendo's game console, Apple's music player (Ipod) or Palm's PDA.

This industry is characterized by the same economies of scale as in the computer field: fixed and sunk costs (plant and products development), and the price bargaining power with the component producers.

If all firms sell hardware products in specialized mass markets, their business models are rather different. On the one hand, 'players' producers (games and multimedia players) often also sell content products (games, music on proprietary standards) and can rely on cross-subsidy strategies. On the other hand, the personal communication tools (PCT) producers (PDA, mobile phones) mainly draw their earnings from hardware sales, even if they generally also sell accessories and software.

Thus, the competitive advantage is slightly different: ergonomic factors and performance for both types, but above all the content portfolio and the installed base on the one hand (player sellers) and the ergonomics factors (easiness of use) and the functionalities available on the product on the other.

So, the regime of competition is rather based on vertical differentiation (console efficiency and the size of the content portfolio) for player sellers while it is rather characterized by horizontal differentiation (with hedonistic prices) for PCT producers .

As they base their commercial strategy on customers locking through software incompatibility (proprietary video games or music cannot be played on other platforms), “player” producers defend strong IP protection[24]. On the contrary, PCT suppliers have begun experiments with FLOSS products since a couple of years:

-Nokia sells an Internet tablet based on Linux and a development community[25],

-PDA Operating system editor Palmsource is working on the integration of its product on a Linux kernel[26].

So, there are incentives to use FLOSS, outside the core competence sphere of the producers (e.g. ergonomics), for cost and compatibility purposes. This also allows better feed back from mobile device advanced users who may develop new features. But incentives to contribute to development, apart from some hard-soft (drivers) compatibility, remain limited. The involvement of those actors in FLOSS efforts should not have an important impact on the development organization, but may strengthen FLOSS diffusion and its de facto standard role.

4.3. Software producers and editors.

Since the beginning of the 1980s some firms have specialized in software production and edition. Basing their earnings on selling licenses, they might be seen as having the greatest interest in opposing the FLOSS model.

---

[23] HP proposes RedHat Linux on his enterprise workstation, but not in its home office solution. However, some minor actors competiting on price use Linux to propose under $300 PC See http://www.silicon.fr/articles/16046/Tribune-PC-a-moins-de-300-merci-Linux.html.

[24] Even if Sony PS2 can possibly run Linux, Sony having disclosed the specifications of its microprocessor to the Linux community (early 2000s)

[25] Nokia 770 Internet Tablet: http://www.nokiausa.com/770/1,7841,feat:1,00.html. Development community: http://www.maemo.org/

[26] http://www.palmsource.com/opensource/.

They are in direct competition from this new way of developing software. Nevertheless a growing number among them are turning to a decisive involvement in FLOSS development. This is what we aim to understand now. For the needs of the analysis we will distinguish operating systems editors from more specialized program editors, as operating systems are the key components for interoperability.

4.3.1. Technical software editors.

Information technologies infrastructure is composed of the combination of a plurality of 'technical' software programs, like data bases, Internet (Web, email) servers, development tools, dedicated application software, etc. They are called "service program" and constitute a "middleware architecture", between the server hardware and the client applications[27].

Archetypes of such firms involved in technical software production are Oracle, MySQL, ACT (Ada language compiler editor), Zope corp or Ilog. They sell technical software solutions, and additionally to the usual economies of scale, the technical aspect of the product induces important learning effects, both on the producer and on the user side.

Letting aside products sales, earnings are increasingly generated by the supplying "3A" complementary services:

- assurance (quality, interoperability, stability),
- adaptation to users' needs and businesses,
- assistance to the usage.

In short, these firms sell services of "maintained, available technical capabilities" (Gadray, 1998) for sophisticated users. Some of them are billing such services via license fees (plus a la carte services), others distribute the product under FLOS licenses and sell only services. All of them are specialized in a single product which corresponds to their core competence. The sector is highly fragmented in small oligopolistic markets for each technical need (for example data base software) with a small number of competing solutions, each one supported by a single firm or, less often, a consortium, like Zope or ObjectWeb software.

Since the 1990s companies in the sector have been at the origin of important initiatives and innovations in terms of IPR management with diversified issues: although some of them remain close to the classical licensing business (Oracle), others have decided to adopt a full GPL strategy (ACT), while others propose a FLOS base with private add-ons (Zope Corp), or a double licensing system combining FLOS and private license for specific clients and purposes (MySQL).

Although counterintuitive it is probably in this domain that incentives to use and contribute to FLOSS are the strongest:

–first, there are historical reasons. Most of Internet infrastructure programs, developed firstly in the university, have been since the beginning protected by FLOS licenses (example: Sendmail, Apache),

–second, there are marketing and competing reasons. In some markets like data-bases there was a dominant actor. For newcomers like MySQL in the data-base market, FLOS was a means to attract potential users and so to circumvent the entry barriers related to the installed base. This strategy has become feasible because of the characteristics of the users: computer professionals (VH users), able to make technical evaluation of the product, to make trials and tests, but also aware of FLOSS principles and in connection with open source organization (via mailing lists, program repository information, contribution to some projects...) On these markets, reputation being above all that of the programs, firms can build their own reputation by diffusing these programs to their clients in order to be recognized as their inventors. Finally, there are solid business reasons. Openness appears as the best possible signal to guarantee standards conformity and compatibility to these advanced and value-added users, giving them access to the source code, and the ability to check how the tool works. As these technical software programs constitute the basic components for any information technologies infrastructure, they must be perfectly mastered by their users. In this context, the demand in

[27] "There is a significant shift underway in the world of software toward what is called service-oriented architecture (SOA), which allows companies to be much more flexible and responsive. As the worldwide leader in middleware, IBM is in a strong position to capitalize on the SOA market" (IBM annual report, 2005, p. 4).

terms of quality, quality assurance, and standards conformity is highly relevant and the signal sent by the openness highly worthy.

Nevertheless, the business remains rather close to that of proprietary software-tool builders, like Oracle or Ilog. The evolution towards FLOSS comes with the evolution towards the production of components. Practically, all these "component producers" have to face a delicate commercial challenge. The most important part of their commercial benefits comes from users' support and components adaptation to final users' needs or to other "utilization technology" producers. This service offer must be clearly defined in order to transform a significant part of these program users into potential clients. Benefits from the joint-services activities have to balance development and maintenance costs, especially those of FLOSS program development, on which they ground their offer.

Therefore, the objective is to transform a handicap (significant investments) into a commercial advantage, by increasing the business feed back from users and considering openness as a way to reduce transaction costs and as a signal of quality. Actually, the main evolution for those firms is to switch from a demand pull strategy (functionality are developed to stimulate/create the demand) to an 'on-demand' development (development when required and paid for or done by the users).

In this sector, the FLOSS producer(s) control(s) development, and manage users' contribution. If some individual contributor becomes important (in terms of contribution volume/quality/innovative aspect), s/he will be hired by the producer(s), with reduced recruitment costs and risks (ACT or MySQL are using this method).

As it is an evolution of the dominant design, if the users' demand remains strong, the FLOSS strategy should diffuse in many technical software sectors.

4.3.2. Operating system editors.

Operating System editors differ from the latter in so far as they specialize in the assembling of multiple software programs into one operating system. So they are in a way bridging computer manufacturers and application software editors, by supplying a key component of the information system that defines its working standard.

Archetypes of such firms are Microsoft, SCO (Unix editor), Mandriva (merger of Mandrakesoft and Conectiva), or SuSE before being absorbed by Novell. The two latter are respectively French and German Linux distributors.

They sell a product (most often bundled with the machine, thanks to Original Equipment Manufacturer – OEM- agreements), but also joined services (documentation, hotline, update/quality insurance). Once again, besides the classical economies of scale, the technical aspect of the product induces important learning effects, on the producer side (development) and on the user side (usage). But the main increasing return to adoption is probably the technical interrelatedness, generating heavy switching costs for changing from one operating system to another and attaching the OS value to the extent of its compatible application software portfolio.

In such a mass market, the competitive advantage is above all the installed base, then the distribution channels (OEM agreements), the brand reputation and the technical staff.

The sector (at least the in the micro-computer field) is a quasi-monopoly controlled by a dominant leader (Microsoft) that intends to use this dominant position in order to extend its dominance on related application tools.

Newcomers entered the market in the micro-computer fields, in the middle of the 90s', as GNU/Linux distribution editor. They sell a commodity (CDRom facilitating the installation of a GNU/Linux distribution via technical add-ons), and joined services (hotline, update). Such a strategy can be understood as vertical differentiation, addressed to VH users and "intensive frontier users" interested in testing this new operating system. This sector works as an asymmetric oligopoly (RedHat is the leader) with a partial geographical segmentation (Mandriva in France and Brazil, SuSE in Germany...)

For these newcomers the incentive to use FLOSS was to capitalize on the Linux user base and existing software, thus to enter in this market at a reasonable price, with an already existing installed base. They have to participate in Linux developments, for technical needs (to develop their absorptive capacity, to have a rapid

access to the last updates, bug fixes…), but also to build their own reputation (vis-à-vis their clients to whom they sell quality insurance and the developers' community). y Their GPL Linux status means they have to publish any modifications under GPL.

Actually, FLOSS distribution does not constitute for them a significant and durable source of income (or, more exactly of profit, as the distributors hold back the main part of the payoffs), but is supposed to generate a demand for maintenance services. As for traditional editors like Microsoft, developing a distribution trademark is of utmost importance since it underlines the quality of the products. It can also open up a wider market share towards "naïve" or at least non-expert users, to whom they can offer value-added services, then improving their related "ARPU"[28]. Because of standardization, these offers will probably be limited to some main distributors, even if they may gradually become available through several local editors/distributors. But this market will remain limited until these products are installed by computer constructors on workstation and end-users machines.

Two main strategies have been put into place to ensure income source of growth: some editors have dedicated their distributions towards organizations sector (Red Hat), other have developed a general public service offer (Mandriva has introduced "users clubs" in which a fixed yearly subscription gives access to various on-line services and software programs). Both strategies contain strong guarantees, after-sales assurance through standard assembling of software programs and standard assistance services.

In both cases, distribution remains the main asset, and these companies growingly "intervene" in Linux development, thus increasing its consortium orientation.

Bringing together SuSE and Novell, or, in the past, Mandriva and Sun (via the distribution of Star Office), reveals that constructors and traditional actors are starting to invest in this field in order to develop the service part of their incomes. This is not totally a new market for them since they already distribute Unix versions and offer a mix of products and services. But the consequence is to reduce the still open opportunities for independent editors.

4.4. Services companies.

A lot of very diversified service companies in terms of size, geographical area or customers are helping users to integrate IT into their activities and provide them with maintenance and assistance, As we cannot analyze each specific model of such companies, we will focus on two polar cases: the global service company, such as Cap Gemini, or the new Novell (or also IBM) aiming to manage big organizations information system, and the small one, specialized, either geographically (in a city or a region) or in terms of customers activity (for instance, in solutions for the food industry).

They do not sell the same products, nor to the same clients:

- the global service company sells global information system solution to IT division in large or medium size companies and administrations, thus to VH clients[29],
- the small service company offers more dedicated solutions (at least on a smaller scale), to SMEs or corporate divisions, at local or sectoral level, thus to clients of very heterogeneous competences regarding IT. They are often also, resellers for server producers.

Both types of firms draw their earnings from 3A services (assurance, adaptation to the user needs and business, assistance to the user) and have to manage increasing return to adoption due to learning effects. But they do not address the same "problem": for global companies, this is a 3A service on a (global) system, granting a high level of availability and efficiency[30], also called "SLA, for Service Level Agreement" in the telecommunications industry, while small service companies sell more localized or specific solutions.

[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] "Capgemini's mission is to support its clients as they transform their businesses in order to improve performance. Located in thirty countries, employing 61,000 people, generating revenues of nearly 7 billion Euros in 2005, the Group offers a wide range of integrated services, coordinated around its four disciplines and an array of sector expertise. These services stretch from strategy making to maintenance of information systems". (CapGemini 2005 annual report: http://www.capgemini.com/annual-report/2005/detail.php?cat=26)

So, the competitive advantage is based, on one hand, on the know-how of managing such global, complex big projects, the brand reputation, and the installed base and, on the other hand, on proximity, related either to industrial business or to geographical location.

This leads to rather different regimes of competition:

- due to the size of the projects, the global services market is rather a "coopetitive" oligopoly. Competitive to acquire contracts, but cooperative as the solution must be open to clients, providers, specific add-ons, and thus abide by standards. These firms cooperate on the definition of the standards, like XML data exchange format[31],
- proximity market rather works as "niches", the intensity of competition depends on the degree of specialization required for each specific business market, and of the number of firms active on the local market.

Concerning IPR status, the situation in services is somewhat strange in so far as, most of the time, the client is the owner of developments made by the service company. So these firms are quite "agnostic" regarding this aspect.

But in both cases, the main reason to use FLOSS is clear: as these products are open (and modular) it is easier to adapt them to client's needs. Using FLOSS for standard components (OS, network, Internet server software), which are seen as "industrial public goods", facilitates compatibility among the different components, the control of the system evolution and can reduce licenses fees. As corporate clients of global services, firms begin to ask for FLOSS products which permits a lower dependence on software editors and helps avoid competitors' dominating standard, while fostering VH client feedback[32].

Incentives to contribute are weak for proximity service companies and depend on their degree of specialization and the innovation dynamics (the more specialized the programs are, the more crucial it is to cling to the technical evolutions - but with less users feed back than in technical software case). For global companies, there are stronger incentives to contribute. It is a way to have an influence on standard settings, but also to integrate users' requirements into software standard versions. To respect their SLAs, they need to be informed of the software evolution (bugs, bug correction, new features...) as quickly as possible. Still following Cohen and Levinthal (1989) analysis, contribution increases the absorptive capacities. And, maybe less important, in a VH user market, it is also a means to communicate on their technical performances. So, if proximity firms do not impact FLOSS organization, as for server constructors, the implication of such global service companies in the development may lead to close development consortia, excluding, or at least making it harder for new developers to contribute.

[30] "IBM uses the cash from its reliable annuity businesses to fund investment in high-value integrated solutions: offerings that integrate services and technology to solve a business or infrastructure problem. Clients increasingly seek solutions rather than "point-product" purchases of particular technologies and products." (IBM 2005 annual report, p. 5).

[31] See the members of the W3 consortium, in charge of editing the Web norms: http://www.w3.org/Consortium/Member/List

[32] Some newcomers have entered the market specializing in services based on FLOSS (and called themselves FLOSS Service Companies). See Jullien (2003, 2005) for an analysis of the marketing strategy of such companies.

	Hardware Component Producers	**Hardware constructors**				
		Server constructors	**Micro computers (Laptops) constructors**		**Dedicated products.**	
Archetype	Intel, ATI	SUN	Segmentation between a quality driven market (« quality laptop », Dell) and a price driven market (« low price laptops » Acer)		Nintendo, Palm, Nokia. Difference between producers of 'player' system and of personal communication tools	
Product	Hardware component. high-performance of use	Hardware and high-performance of use (characteristic of use)	Hard + high-performance	Hard + easiness of use	Video game console, music players	PDA, mobile phone
Users	Hardware constructors. Sophisticated users may be prescriptors.	Sophisticated users. VH++	Intensive frontier users (B. Kogut & A.Metiu)	Mass market	Mass market	Mass market
Earnings, added value	Product	Hardware, maintenance, assistance/hotline. (HMA model)	Hard, differentiated	Hard in volume, standard	Integrated product + digital content (game, music)	Integrated product + application software (emerging)
Competitive advantage	Component high-performance, price, industrial capacity, brand	Brand Reputation Installed base Distribution channel technical staff Hardware high-performance application portfolio	Quality price ratio (price of the high-performance) Reputation, brand	Price. Distribution channel	ergonomic factors, high-performance Content portfolio	ergonomic factors, high-performance Complementary features
Competition regime	(stable) oligopoly.	(stable) Oligopoly. Market power. Technological lock-in (high switching costs) Innovative capacity Cournot + quality	Horizontal differentiation Quality	Price + market size	Vertical differentiation (high-performance, content port folio)	Horizontal differentiation with hedonistic prices
IPR management	N/A	In transition from close to open source	None	none	None. Strong IP protection	Few evolution. Strong IP protection
Incentives to use FLOSS	Compatibility of the component with FLOSS	1) incumbents: Cheap answer to newcomers (PC servers with	FLOSS quality and features	FLOSS price	Outside the core competence sphere, integration of	Outside the competence sphere, integration of

Table 1. Business models and attitudes towards FLOSS.

	market (mainly Operating system)	Microsoft NT solutions, DELL) 2) all: MS-NT competition.			complementary features (Internet connexion), the price games: OS no	complementary features (Internet connexion) user development integration (photos, email access, etc.) PdA, mobile phone, OS possible
Emergence of the integration of FLOSS into the products	00'	00s	End of 00'	End of 00'	End of 00'	End of 00'
Incentives to contribute to FLOSS projects	Development of product drivers if the market is big enough	Hard-soft compatibility, users requirements, quality insurance	Coming. Hard-soft compatibility (drivers), quality insurance	None	Weak	Strong (hard soft compatibility)
Influence on the FLOSS development organization	None	In transition from internal to consortia.		Not relevant	None.	No influence. Following the existing organization.
Future evolution	Following the diffusion of FLOSS.	Consolidation/ stabilization	Emergence	Emergence	Emergence on very restricted features	Emergence on an open embedded OS.

	Software producers-editors			Service company	
	OS producers & editors.		**Software Components & tools producers**	**Information System assemblers 'architecters'.**	**Small service companies (SSLL or SSII)**
Archetype	Segmentation between OS producers and editors (Microsoft, SCO) and newcomers, using FLOSS operating system (Linux) to enter the market via distribution edition (RedHat, Mandriva, ex SuSE)		MySQL, ACT, Zope firms / Oracle	Cap Gemini, Novell,(but also the new RedHat, and more and more IBM (IBM Global services) or HP)	N/P
Product	Own Operating System	Linux distribution	Technical software solution	Global Information System solution	possibly dedicated IT solution (hard + soft)
Users	Mass market	»Enlightened » general public	Sophisticated users. VH++	Large firms and organization including sophisticated users. VH++	Naïve
Earnings, added value	licenses	Commodity (CD, utilities and tutorial) Joined services (hotline, update) PHU model	3A service on a product Assurance (quality), Adaptation to the user needs and business Assistance to the use	3A service on a system Assurance (quality), Adaptation to the user needs and business Assistance to the use	Customized 3A service Assurance (quality), Adaptation to the user needs and business Assistance to the use
Competitive advantage	Installed base Brand Reputation Distribution channel	Brand Reputation Installed base Distribution channel technical staff	Core competence product	Know-how, experience, Brand reputation, installed base	Industry business knowledge (cognitive proximity), geographical proximity
Competition regime	Quasi Monopolistic competition	Partial geographical segmentation. Asymmetric oligopoly	Oligopoly competition	Coopetitive Oligopoly.	Variable on proximity market.
IPR management	Strong IP protection	GPL kernel and complementary components (mainly GPL ones, but not exclusively)	Fine tuned IP management (Hybridization: double licensing, open based and proprietary developments)...	Hybridization. Agnostic. Important role of the standardization game	N/R

Table 1. Business models and attitudes towards FLOSS.

Incentives to use FLOSS	Few. Outside the core competence sphere, integration of complementary features	De facto	Signal to client (respect of the standards, compatibility), standard effect (diffusion and test are possible) better feed-back management (externalization of a part of the R&D)	To avoid competitor's dominating standard. Strong innovative potential. Reliably to users requirements/needs	Cost, quality and reliability.
Emergence of the integration of FLOSS into the products	Not relevant	90s	90s	00'	00'
Incentives to contribute to FLOSS projects	Few. Technical needs; absorptive capacity.	Technical needs: absorptive capacity, quick access to the last updates. Construction of the reputation (clients and community) Legal commitments (License)	Technical needs: absorptive capacity, quick access to the last updates. Construction of the reputation (clients and community) Legal commitments The firm is linked to the product. (License)	Contribution to industrial public goods developments influence on standard settings Integration of users requirements into the standard version of the piece of software Absorptive capacities	Weak. Depending on innovation dynamics, and degree of specialization
Influence on the FLOSS development organization	Not relevant	Mix: community + in house development.	The firm/consortium controls the development. Users are contributors, and skilled contributors are hired by the firm/consortium	Targeted communities. Risk: the standard game may lead to a close consortium excluding new developers contributions.	No influence. As it is.
Future evolution	Possible marginalization. Evolution to an open strategy.	Declining? RedHat vs Ubuntu -> supply by hardware company -> market growth rate	Possible extension to a large set/scope of technical components producer.	Dominant design for large organizations.	growth

5. Business models and attitudes towards FLOSS, A synthesis.

From this description is is now possible to infer an initial characterization and explanation about the variety of observable behaviors towards FLOSS. First of all it is important to notice that actors in all the segments of the IT industry have begun to take an interest in open source software; in other respects, firms from any of these segments seem to conform to very similar patterns. This section aims to better understand the origin and motivation of those attitudes following the characteristics of the concerned segments. However, we will leave aside two segments that seem us not to play an important role in this scenario: components producers and small services companies. Components companies because they only can have very reduced involvement in FLOSS development, limited to the drivers they provide for each operating system of sufficient market share. Small services companies, as far as they are concerned, represent a small and disparate segment unable to impact significantly FLOSS development and organization. Regarding FLOSS it is reasonable to consider they do not actually distinguish themselves from independent developers.

Leaving these two segments aside it is then clear that, first of all, the level of commitment to FLOSS dynamics varies greatly , from a possible driving role to a weak or even inexistent involvement. Additionally we will show how the observable characteristic patterns and strategies in every considered segment can be explained as derived from the competition regime and the type of users concerned

Regarding the degree of involvement in FLOSS dynamics a first remark is related to the significance of the diagonal in the following table. It emphasizes that the driving forces and more active actors are those deeply engaged in software development and use either as a core activity or as a crucial condition of hardware performances, as it is the case for server manufacturers deeply involved in the Unix world. In the opposite position are hardware suppliers that can only feel concerned by FLOSS for compatibility and price purposes. Even if those latter do not have to play an active role in FLOSS development and organization, their existence and strategic choices remain of importance in so far as this conditions the evolution of FLOSS market share, thus future perspectives within the software industry.

Table 2. Main actors and degree of involvement in FLOSS dynamics.

Degree of involvement	***Driving force***	***Active***	***Weak (compatibility)***	***Weak to nil***
Software	Technical software[1] and architecturers[2]	OS Editors		
Hardware		Servers	HQL[3] and PCT[4] Players (potentially)	LPL[5]

[1] Software Components and tools producers; [2] Information Systems Assemblers; [3] High Quality Laptops; [4] Personal Communication Tools; [5] Low Price Laptops

However, in order to further analyse their postures, it is necessary to examine the competition regimes they face as well as the types of users that make up their markets.

The regime of competition.

All these segments are characterized by strong imperfect competition regimes due to diverse increasing return effects (economies of scale in production, high sunk costs – R&D, distribution channels... -, technological interrelatedness and learning). One single segment (OS for PC computers) can be considered as a quasi-monopoly with very high barriers to entry.

In these oligopolies, differentiation strategies widely play an important role. In the sole segment of low-price laptops, where competition is mainly based on prices, the strategies are built on the exploitation of new market potential trough vertical differentiation with the traditional PC market. In the other segments, strategies are rather of horizontal differentiation either related to the integration of new features and high performance tools or to market segmentation through hard-soft-content bundling (mainly based on proprietary standards).

So new entries require either better performances/cost ratios (for instance lower price laptops or better computation capacities for servers or high quality laptop) or innovative horizontal differentiation (e.g. new feature in personal communication tools or technical software tools). But in segments like players hard-soft-content bundling strategies make it very expensive to enter the market without perspectives of open standards. In order to capture a significant market share, a newcomer should add to his device offer (for instance a new game console) a content portfolio large enough to attract new customers or to make former customers likely to switch from their earlier provider. In thePC market however, IBM's open strategy at the beginning of the 80's, debundling hardware from OS, made it possible to enter the hardware segment. Last but not least, horizontal differentiation can also target a sub-group of users, with specific needs, not fulfilled by the existing offer, as Samsung did when proposing a fold display mobile phone.

The users.

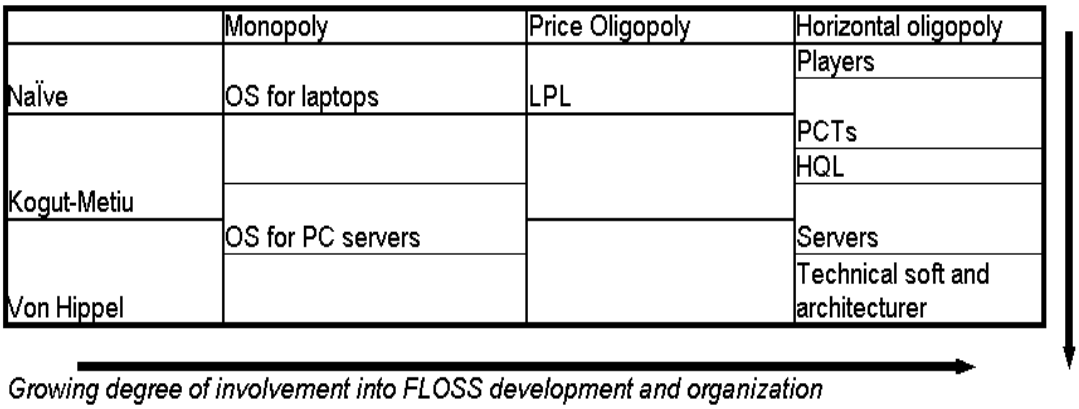
Users have a double role to play, that derives from both their economic and technical standing, Traditionally, user market power is a function of its buying capacity. At one extreme, clients and contracts in the "architecture" market are large and generally endowed with significant technical competencies. So they are likely to influence economic and technical choices. At the other extreme low price laptops address a mass market where individual users have very little budget and few skills. Their influence on market evolution is negligible at an individual level but of global importance in terms of elasticity to prices.

Our analysis is mainly based on a typology of users following the type and growing level of their competences. Three main types have to be distinguished. The first is the category of "Naïve users" (N) that are not endowed with noticeable technical skills and do not weigh very much in economic terms. The second is the category of "Kogut-Metiu Users" (KM) that are not able to contribute to software development but can originate new features or innovations by revealing their own needs and above all represent an irreplaceable testing and debugging base. KM users are sensitive to price and quality arguments The third category is that of the "Von Hippel Users" (VH) that act as "sources of innovation" able to contribute to software development by proposing improvements or modifications.

Attitudes towards FLOSS

It is thus stimulating to see how crossing the type of users/clients with the type of competitive regimes can discriminate the listed sub-markets and to see how this interbreeding can enlighten a growing involvement in FLOSS development and organization for commercial actors.

Table 3. IT sub-markets following the nature of users and competition regimes.



Linux distribution editors have been among the first commercial actors to enter the market using FLOSS. This could be seen as obvious on a mass market with rather naive users and a significant prices based competition. However, consumers buy computers with an already installed OS and few of them are skilled enough to install a different OS. Additionally there are no incentives to do it because the first installed OS has been already paid

for with the computer. On the emerging PC server market things work differently. Most if the users, of VH or KM type, are aware of the technical questions for installing and configuring an OS. FLOSS give them access to a cheaper but also more open and more adaptable Unix-like operating system, than they could find in the traditional Unix offer. This gives FLOSS OS editors an undeniable competitive advantage.

What now needs to be taken into account is the possible changing attitude of laptop producers towards the dominant position of Microsoft in the OS market. First, in the segment of low price laptops (LPL) where users are mainly naive, competition is overall based on prices. Installing a FLOSS OS can be considered as a way to reduce price, hence to improve the firm's market share, to an enlarged market or at least to permit a new entry by compensating not yet accessible economies of scale. In the high quality laptops (HQL) things work differently as we will see below.

When looking at the "horizontal oligopoly" column, it is clear that the more skilled users are, the more FLOSS concepts and industrial offers are likely to spread.

At one extreme, in the games consoles segment but also to a lesser extent in the music player market, proprietary formats have introduced, as seen above, a strong bundle on hardware-software-content. Thanks to the MP3 standard or new open existing or emerging standards like ogg, new entries are always possible on segments like the music players market. On the contrary, barriers remain high on the video game players market due to the scarcity of independent games liable to run on Linux, unlike the PS2, Xbox and other proprietary standards games. Moreover, when they exist, such games seem harder to obtain for naive users.

The opposite polar case is the servers market, where producers were used to providing proprietary solutions with proprietary Unix. Here suppliers have to deal with high skilled VH users that can be of essential contribution in a context of FLOSS opening. The rise of PC servers has permitted some users to avoid such a bundling problem; moreover, using Linux allows a cheaper offer (vertical advantage) reusing Unix programs (content) portfolio. Thus some firms have been able to enlarge the servers market from VH users likely to manage their systems by themselves to KM users, sensitive to prices, but also to the quality of a PC server fitted out with Linux. So new entries have been experienced like the Cobalt[33] one, but the main actors of the Unix "world" have also rapidly developed their own offers, cutting down the sources of vertical differentiation.

Personal communication tools and high quality laptops represent an intermediate case with less skilled users (KM+N) and a weak degree of involvement of commercial actors only motivated by preoccupations of compatibility and absorptive capacity.

From the beginning of the year 2000, PCTs architectures and functions have been opened in terms of potential applications and external interoperability. Operating systems are no longer at the heart of the products differentiation that is rather based on ergonomic aspects and hardware characteristics. Thus in the absence of a still established de-facto standard as it stands in the PC market, Linux can be considered as a rational choice for PCT suppliers, in so far as it is free of charge and benefits from a community of users able to develop new features and new products out of any proprietary control. This, to implement Linux on PCT devices appears as a good strategy in order to limit differentiation to the core competences of the manufacturers (content products not being controlled by a firm).

On the contrary, there is a de-facto standard in the HQL market (Windows products). HQL producers may find it hard to switch from Windows to Linux, because this would mean either acquiring new skills (OS management and improvement), or sub-contracting this maintenance to Linux editors (RedHat, SuSE,...) which may lead to another dependence and to difficult relations with the dominant provider. Nevertheless, a possible future evolution in this sense is likely to arise from the pressure of users becoming more aware of the potentialities of switching to FLOSS. In the near future some of the HQL will probably switch to debundle their machines from the associated OS, giving the KM users the choice between a Windows and a Linux platform.

Finally, in the markets where users are mainly of VH type (technical software and architectures) many firms have already turned to FLOSS or are about to do it. This represents an effective means to vertically differentiate their offer in terms of performances (technical qualities of the product, better attention to users' needs and feedback...) These strategies have been initiated by firms that didn't occupy a dominant position in

[33] Cobalt was bought by SUN, which dissolved the products into its own offer. See http://www.sun.com/hardware/serverappliances/eol.html

their markets, either newcomers (like MySQL on database market) or in business troubles (like IBM with its Eclipse development tool). This FLOSS strategy therefore allowed those firms to reduce their development costs, by integrating the contributions of VH users, but also to decrease the marketing costs by taking advantage of the diffusion dynamics within the population of skilled, VH or KM, users.

So this short analysis shows that users occupy a crucial place in the understanding of firms involvement in FLOSS development and organization, as summarized in Table 3. All things considered, this is not so surprising in so far as the very nature and origin of the open source concept is the willingness of skilled users to better satisfy their own needs and to share the result of their efforts. This remains unquestionable.

Tableau 4. The decisive role of the users.

Users type	***Competition regime***	***Actors/ « products »***	***Degree of involvement***	***Motivations to FLOSS***	***Aim***
VH	Horiz. Different. Oligopoly	Technical software and architecturers	Driving force	Intrinsic and sustained quality + signaling	Entry (My SQL) Market share (vertical diff.)
VH	« Split » Oligopoly	Servers manufacturers	Active	Quality and openness within Unix world	Response to the PC-servers rise
KM	Strong Monopoly	OS editors	Active	Unix Achievements + Open standard	Barriers to entry bypassing
KM	Horiz. Different. Oligopoly	High Quality Laptops suppliers	Weak (absor-ptive capacity)	Quality + open application tools	Vertical differenciation (price/performances)
KM+N	Horiz. Different. Oligopoly	PCT	Weak (compatibility)	Additive Features	Limit differenciation to core compentencies
N *	« Split » Oligopoly	Players	Weak (compatibility)	Debundling hard-soft-content	Barriers to entry bypassing
N	Price based oligopoly	Low Price Laptops suppliers	Weak to Nil	Prix	Price competition

* but VH necessity for open content

6. Conclusion: The critical role of users.

So as far as FLOSS adoption is related to marginal aspects of differentiation, it has little impact on the industrial structure and competition. This is generally the case for most of hardware producers, when hard-soft-content is not bundled any more (servers, laptops, PCTs, DVD and MP3 players...) Their core competences (and hence the barriers to entry) are in the order of the product design (ergonomic, hardware efficiency) and the management (production, logistic and marketing) rather than in software development. FLOSS evolution does not take part in their competitive environment and it becomes clear for them to integrate external FLOSS components as soon as they are available and can be helpful to their customers. The situation is of course different in the segments not yet unbundled, such as game players. Content remains an important source of cashflow and content development remains a core competence protected by the proprietary nature of the operating systems.

So IP regime can be opened without calling the firm's competitive advantage into question in so far as it remains out of its core competences. The latter would meanwhile benefit from a strong IP protection regime likely to defend its market position.

But how does this assertion apply to non-hardware firms? As seen before, their core competences have evolved and significantly shifted. The main challenge in the computer industry is less and less to supply a solution to a given problem at a given time, but increasinglyy deals with short to long term uncertainty regarding IT system production and management. Users ask for solutions able to protect them against uncertainty, granting interoperability, bugs resolution, new needs satisfaction and technical evolutions integration. The trade off between available solutions is not posed in terms of their cost of acquisition but of their "TCO" (total cost of ownership), into which the future costs and the costs for granting interoperability and adaptability have to be estimated. This is precisely what architecturers, technical programs and OS producers sell to VH and KM users, aware of these problems and signals. On these markets the FLOSS organization represents an asset for producers, who can claim their involvement and succeed in building, sustainable business models.

In such conditions, the open IP regime can be seen as a very efficient solution to the schumpeterian dilemma in so far as it permits a wide diffusion of knowledge, while encouraging innovation, as producers are incited to contribute to the development of the product they use/sell.

In each of these markets, all the recent new entries have been based on the competitive advantage drawn from the FLOSS label: FLOSS OS editors (like RedHat), FLOSS database producer (MySQL), FLOSS service companies (VA Linux, or Linagora in France). Today incumbents are also assuming this strategy (IBM with Eclipse, SAP opening its data base system, even Microsoft opening some of its technical tools...) This could shortly become the benchmark of industrial organization on these markets, inducing a growing control of FLOSS development by commercial firms and a spectacular enlargement of open IP regimes in the software field.

References.

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

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

D; Demazière, F. Horn and N. Jullien, 2006, How free software developers work. The mobilization of "distant communities", working paper, M@rsouin. http://www.marsouin.org/articles.php3?id_article=116. A former version (in French) has been published in Cahier lillois d'économie et de sociologie, 2005, n°46, pp 171-194.

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

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.

D. Foray, S. Thoron and J-B. Zimmermann, 2006, Open Software, Knowledge Openness and Cooperation in Cyberspace, forthcoming in Brousseau E. et Curien N. *The Economics of Internet*, Cambridge University Press.

L-A. Gérard-Varet and J-B. Zimmermann, 1985, Concept de produit informatique et comportement des agents de l'industrie, Colloque "Structures Economiques et Econométrie" - Lyon, 23 - 24 mai 1985 .

M. Hapke, N. Jullien et J-B. Zimmermann, 2005, Does using libre software imply contributing to its development?, CALIBRE Workshop, Paris, 4th March 2005, http://www.calibre.ie/paris/

IDA/GPOSS, 2004, Advice report - EUROPEAN COMMISSION / Enterprise Directorate General - IDA/GPOSS - Encouraging Good Practice in the use of Open Source Software in Public Administrations -

Report on “Open Source Licensing of software developed by he European Commission (applied to the CIRCA solution)” http://europa.eu.int/idabc/servlets/Doc?id=21197

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

N. Jullien, 2005. Firms' strategies facing FLOSS diffusion. Upgrade, The European Journal for the Informatics Professional. Vol. VI, issue no. 3 (June 2005)

N. Jullien & **J.-B. Zimmermann,** 2006a, New Approaches to Intellectual Property : from Open Software to Knowledge Based Industrial Activities, forthcoming in Patrizio Bianchi and Sandrine Labory (Eds.) *INTERNATIONAL HANDBOOK OF INDUSTRIAL POLICY,* Edward Elgar 2006. http://www.vcharite.univ-mrs.fr/greqam/pdf/working_papers/2005/2005-39.pdf

N. Jullien & J.-B. Zimmermann, 2006b, "Peut-on envisager une écologie du logiciel libre favorable aux nuls ?", to be published in Terminal. http://www.marsouin.org/article.php3?id_article=65

B. M. Kogut and A. Metiu, 2001. Open Source Software Development and Distributed Innovation, Oxford Review of Economic Policy, Vol. 17, No. 2, Summer 2001.

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.

K. Lakhani et R. Wolf, 2003, Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects, MIT Sloan School of Management Working Paper No. 4425-03.

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

J-P.Smets-Solanes et B.Faucon ,1999, *Logiciels libres - Liberté, égalité, business*, Edispher, Paris.

J.-B. Soufron and J. Sallantin, 2005, The evolution of free/libre and open source software licenses : a dynamic model, communication to the European Conference on Complex Systems, Paris, Nov. 14-18.

M. Vivant, 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, 2002, Open Source Software as horizontal innovation networks – by and for users, MIT Sloan School of Management W.P. N°4366-02.

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

J.-B. Zimmermann, 1998, L'industrie du logiciel: ébauche d'une approche prospective, *Terminal* Hiver 97 - Printemps 98, N°75.