

Free/Libre/Open Source Software (FLOSS):
lessons for intellectual property rights management
in a knowledge-based economy.

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

July 27, 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 to draw benefits from such involvement and their connected activities. In other way we want to stress the different types of business model these firms are leaning on and the possible evolution they are likely to know in a 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 of a typology of the different business model 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 intent to 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

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. Copyright protecting 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 suit 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 the software products interoperability. On the other side 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 imposing 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 raises immediately 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 way 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 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 originated in the initial movement. Such a turn still appears to be part of the actual way of working of FLOSS in so far as a growing amount of the code is produced by salaries that are paid for doing so. Such “hybridization”, mixing market and non-market rationales, nowadays appears as an inescapable evolution that also challenges policy makers for integrating 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 to draw benefits from such involvement and their connected activities. In other way we want to stress the different types of business model these firms are leaning on and the possible evolution they are likely to know in a near future.

Section 2 is devoted to the analysis of IPR traditional protecting ways 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 the 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 the setting up of a typology of the different business model that can be observed in the OS landscape, how they correspond to different strategies of industrial firms

1. Introduction

according to the main characteristics of their technical skills and market position. Finally, in a conclusive section we will intent to 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 lied to the market for computer system. 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 good 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 leaning on the resources of the computer it is implemented in. Alternatively, in the case of a system software, 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 with imperfection 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 bring 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 do 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. The 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 have been granted during the 1990s in the Unites States. In Europe, the European Office of Patents remained on its initial position to grant patents only in the case in which they are an integral component of an industrial device or process.

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

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-art in the field of computer software programming, patents are granted on a totally arbitrary way, giving private rights for the use of resources that had been formerly shared by 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 with 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 that will be able to manage such a situation are the large companies that will have the capacity to build a large portfolio of patents. The cost of the defense will be so high for SMEs that an attack for patent infringement may threaten their survival¹. 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 cost². 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 to IBM and send a letter to 1500 other companies to inform them of the risk they run in the case they continue to offer solutions derived from GNU/Linux³.

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 in the 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.

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](http://www.zdnet.fr/actualites/informatique/0,39040745,2135115,00.htm), « Le camp des logiciels libres dénonce SCO dans sa guerre contre Linux » <http://www.zdnet.fr/actualites/informatique/0,39040745,2135115,00.htm>

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

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 GNU⁴ 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 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 it can 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 principle 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 motivation 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 cooperate⁵.

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 INRIA⁶ with the CEA and the CNRS⁷ have 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 the 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 is not yet 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

⁴GNU's Not Unix

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.

⁶National Institute for Research in Informatics and Automatics.

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

⁸G rard Giraudon, chairman for industrial developments and relations at INRIA, in Y.Rocq “Faut-il adopter la license CeCILL”, Login, n 120, Sept. 2004.

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

will be treated by Paris courts. This represents an important advantage for French developers and enterprises which do not have to take proceedings into a foreign court, even if it may be a brake for involving foreign collaborations⁹. At the European level, a recent report to the European Commission stresses that GPL and other OSS license do not meet the requirements of 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 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)

Taking count of the diversity of the practical ways of adopting the principles of open intellectual property management in the field of software, we will use below the acronym FLOSS (for Free Libre and Open Source Software) that is nowadays widely used when talking of the different approaches of this “Open Source” movement in a single unified category.

3. From an individuals' 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 not any actual economic significance. Things however turned to a radically different situation with the appearance of new adopters, a new demand from simple users that 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 concerned base of users 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 performances and reliability. Secondly this enlargement has been 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, users 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 surf on the Internet. This new category of actors in the FLOSS world found his proper way to reconcile commercial requirements with the principles of non-appropriation by grounding 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 into 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 way, 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 to 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 the one of gathering the relevant information and transmitting it to the developers community¹⁰.

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

¹⁰ See Hapke, Jullien and Zimmermann (2005)

3. From an individuals' benevolent contribution to an industrial based model.

On the contrary other aspects can play a negative role. Of course one can think to the free riding attitude these new users could be accused, as drawing benefit from public goods to whose production they didn't contribute. But this is not a real problem because as 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 none 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 behaviors of opportunistic firms could take shape by drawing private benefits from the marketing of products they didn't contribute to develop.

Of course the developers motivations are not only of 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 couldn't be of no consequence. Voluntary and not too much expecting in terms of revenues, FLOSS developers had then to interact more and more frequently with entrepreneurs with much highest income perspectives. Sometimes the developer becomes entrepreneur, sometimes he is only the supplier of the products that will give rise to the 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 them a renewed importance to utilitarian considerations. It can then lower the developers mobilization hence the efficiency of the cooperative process. As a consequence, being frustrated from this confrontation, some of the developers can be led getting out of the game (Foray and Zimmermann, 2001) or allocating a growing part of their time to more lucrative activities either or not related to the FLOSS world. A better understanding of this question implies to look a while onto 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 to duplicate efforts of software development or improvement. In other word 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-exclusive¹¹. 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 it about an on line system of inter-users assistance. Thanks to the growing storage capacities and transmission potential given by the 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 ethic motivations two other kind 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, what ranks this "extrinsic" feature at the 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 editors¹². Following and taking part to 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

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).

3. From an individuals' benevolent contribution to an industrial based model.

first stems from a learning by doing effect, drawn from 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 advance in the programmer's abilities, larger beyond as what could have been drawn from his involvement in a more traditional development team, even within large enterprises where software production organization remains compartmentalized and divided in small teams. This learning by interacting is of great deal for a programmer in order to hold one's own into 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 former¹³ and work in a similar way as in the academic sphere. It 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 an engagement in an enterprise or a better access to a funding source.

Foray, Thoron and Zimmermann (2006) show that incentives related to learning aspects work differently following 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 consumers heterogeneous from their consent to pay for those services quality, this 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 developers¹⁴. As Demazière & al. (2005) shown, these two roles are not opposite, but represent more two steps, two periods in a developer life (or "career").

Of course, it would be rather naïve to think that a large taste for entrepreneurship from practiced developers would naturally originate in 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 few 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 high 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 F/OSS project."

So enterprises involvement in the FLOSS world appears growingly as a natural substitute to the individuals' voluntary contributions based model. This is all the more important these new FLOSS industrial actors are endowed with a key responsibility for gathering and analysing the huge amount of informations originated in simple users FLOSS products utilisation and transferring it to the developers communities¹⁵.

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.

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).

4. From business models to industry structure.

4. From business models to industry structure.

The central question for economics turns to understand why firms deliberately involve in FLOSS production thus abandoning any monopoly power onto the fruit of their efforts, then any perspective of durable profits by selling those products as such, but in the contrary allowing a free access to them for users and potentially competitors. 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 for doing 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 into the 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 products 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 for 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 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 programs¹⁶.

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 firms' 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 distinct 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 activities, characteristics of the products and characteristics of the users 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 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 important 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

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 stands 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”.

4. From business models to industry structure.

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 FLOSS 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 do propose such compatible drivers.

But, this remains a marginal contribution, and should not have before long any 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 PDA or mobile phone are built to support a growing number of applications¹⁷.

4.2.1. Servers.

These computers are aimed 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 important 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 characteristics of uses (high-performance, reliability, "evolutivity"). The users are computer literate people, what 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 markets 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 good assistance to users). The industry structure is a stable oligopoly with important barriers to entry and decidedly role of quality.

But new strategies, based on more open architectures (PC servers and FLOSS operating systems) are emerging, since the beginning of the 2000s, when IBM, followed by HP announced it will 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 than the other operating systems and thus facilitating its diffusion;

¹⁷ This distinction between specialized and generalist devices is evolving, as Sony aims its PS3 to be the media center at home. But this did not change (for the moment?) the industrial structure.

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

4. From business models to industry structure.

- 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 market¹⁹;
- finally firms whose activity leans on a standard they do not control directly have interest to choose the most open as possible, so as to avoid to depend on the strategy of its owner and to stay aware and involved into 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 the hardware-software compatibility, in a market where performances and sustainability are very important,
- it is the only way to grant to users that their needs will be integrated into the future versions of the FLOSS product,
- as explained by Cohen and Levinthal (1989), the participation to the development increase 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,
- 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 foundations²⁰. 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 from 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 to run 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 to organizations or intensive end-users²¹ and “*low price computers*” -*LPC*-, targeted to 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 products ranges in order to draw benefits from enlarged scale and scope economies. This appears quite similar to other industries like automobile where top- and bottom-of-the-range markets do not work similarly but can be partly held by the same manufacturers.

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 its business model, with recurrent losses.

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 game, watching video films on their computer.

4. From business models to industry structure.

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

Microcomputers industry is characterized by important 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 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 users²²”. On the opposite LPC is a mass market, where users have no particular skills.

Earnings in these markets come from the product sales.

The competitive advantage, additionally to the capillarity of the distribution channel, comes from quality-price ratio for QC manufacturers, and from the price for LPC's. 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 purpose (easiness to develop maintenance and quality insurance services) on QC market, for price purpose (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 can 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.

However, as far as we know, there has not been yet any FLOSS based offer in the microcomputers markets coming from the main players²³, 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 kind of machines, computers, but also dedicated machines as PDA, multimedia players, mobile phones, ... Most of these devices get in connection to the computer(s), but more and more they also access to the Internet (via Wifi, local networks or adsl technologies), and share applications with the computers (email, personal data management, etc.) Even devices like video games consoles are growingly 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 than 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 a 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) draw mainly 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 (players sellers) and the ergonomics factors (easiness of use) and the functionalities available on the product on the other.

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

²³ HP proposes RedHat Linux on his enterprise workstation, but not in its home office solution. However, some minor actors competing 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>.

4. From business models to industry structure.

So, the regime of competition is rather based on vertical differentiation (console efficiency and the size of the content portfolio) for players 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), “players” producers defend strong IP protection²⁴. On the contrary, PCT suppliers are beginning experiments with FLOSS products since a couple of years:

- Nokia sells an Internet tablet based on Linux and a development community²⁵,
- PDA Operating system editor Palmsource is working on an integration of its product on a Linux kernel²⁶.

So, there are incentives to use FLOSS, outside the core competence sphere of the producers (e.g. ergonomics), for cost and compatibility purpose. This also allows better feed back from mobile device advanced users who may develop new features. But incentives to contribute to the development, apart the some hard-soft (drivers) compatibility, remains 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 onto software production and edition. Basing their earnings on selling licenses, they might be seen as having the greatest interests against the FLOSS model. They are the most directly competed actors of this new way of developing software. Nevertheless a growing number among them is turning to a decisive involvement into FLOSS development. That's what we aim to understand now. For the needs of the analyze we will distinguish operating systems editors from more specialized programs editors, as operating systems are the key components for interoperability.

4.3.1. Technical software editors.

Information technologies infrastructure is made by 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 applications²⁷.

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 growingly generated by the supplying “3A” complementary services:

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

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/>.

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).

4. From business models to industry structure.

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 on a single product which corresponds to their core competence. The sector is highly fragmented in small oligopolistic markets for each technical needs (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 enterprises of the sector have been at the origin of important initiatives and innovations in terms of IPR management with diversified issues: if some of them keep close to the classical licensing business (Oracle), some 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 purpose (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, are 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 mean to attract potential users and so to circumvent the entry barriers related to the installed base. This strategy has got 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 before 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 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 is coming 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 backs 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 are done when required and paid or done by the users).

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

4. From business models to industry structure.

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 define its working standard.

Archetypes of such firms are Microsoft, SCO (Unix editor), Mandriva (merge 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 an 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 before 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 of this dominant position in order to extend his dominance on related application tools.

Newcomers entered the market in the micro-computer fields, in the middle of the 90', 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 strategy can be understood as vertical differentiation, addressed to VH users and “intensive frontier users” interested to test 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 Linux user base and existing software, thus to be enter in this market a reasonable price, with an already existing installed base. They have to participate to 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). They are committed by the GPL Linux status 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 points the quality of the products. It can also open them a wider market share towards "naïve" or at least non-experts users, to which 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 will be 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

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.

4. From business models to industry structure.

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 implies strong guarantees, after-sales assurance through standard assembling of software programs and standard assistance services.

In both case, the distribution remains the main asset, and these companies growingly “intervene” in Linux development, increasing the 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 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 opened opportunities for independent editors.

4.4. Services companies.

A lot of service companies very diversified 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 clients²⁹,
- the small service company offers more dedicated solutions (at least at a smaller scale), to SMEs or corporate divisions, at local or sectoral level, thus to client of very heterogeneous competences regarding IT. They are also, in many time 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 use) 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 efficiency³⁰, also called “SLA, for Service Level Agreement” in the telecommunications industry, while small service companies sell more localized or specific solutions.

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, global services market is rather a “cooperative” oligopoly. Competitive to acquire contracts, but cooperative as the solution must be opened to clients, providers, specific add-ons, and thus abide by standards. These firms cooperate on the definition of the standards, like XML data exchange format³¹,

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>)

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>

4. From business models to industry structure.

- proximity market rather work 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 a bit odd in so far as, most of the time, the client is the owner of developments done by the service company. So these firms are quite “agnostic” regarding this aspect.

But in both case, 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 the 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 claim for FLOSS products this permits a lower dependence to software editors and to avoid competitors' dominating standard, while fostering VH clients feedbacks³².

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 backs 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, bugs correction, new features...) as quick as possible. Still following Cohen and Levinthal (1989) analysis, contributing increase the absorptive capacities. And, maybe less important, in a VH user market, it is also a mean to communicate on their technical performances. So, if proximity firms should not impact FLOSS organization, as for server constructors, the implication of such global service companies into the development may lead to close development consortia, excluding, or at least making harder for new developers to contribute.

³² Some newcomers have enter the market specializing themselves into 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) | 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 portfolio) | 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 |

| | | | | | | |
|--|---|--|--|--------------|---|--|
| | 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 enough important. | 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 FLOS 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 | Grand public "éclairé" | Sophisticated users. VH++ | Large firms and organization including sophisticated users. VH++ | Naïve |
| Earnings, added value | licenses | Commodity (CD, utilitaires 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 |
| Incentives to use | Few. Outside the core | De facto | Signal to client (respect of the | To avoid competitor's dominating | Cost, quality and reliability. |

| | | | | | |
|--|--|--|--|--|---|
| FLOSS | competence sphere, integration of complementary features | | standards, compatibility), standard effect (diffusion and test are possible) better feed-back management (externalization of a part of the R&D) | standard. Strong innovative potential. Reliably to users requirements/needs | |
| 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. Conclusion

to be done ...

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](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.