FLOSS in an industrial economics perspective

Nicolas Jullien, Jean-Benoît Zimmermann

FLOSS in an industrial economics perspective.

Nicolas Jullien, Jean-Benoît Zimmermann[1]

Abstract

The spread of free/libre open source software (FLOSS) represents one of the most important developments in the Information Technology (IT) industry in recent years. Within the context of a knowledge-based economy, this sort of approach appears exemplary for a growing number of industrial activities in which the amount of knowledge that has to be mastered is too large for a single agent, however powerful. Considering knowledge as a mutual resource requires a rethinking of the value chain concept, since cash flow is derived from use of the knowledge base (services, complementary products), not from the knowledge itself. In a classical industrial economics perspective, this reshaping of the value chain must be analyzed not only at the global ecosystem level (who produces what, between firms and universities, users and producers, etc.), but also at the industry level (once the industry's role has been identified, how does it organize itself?). Various points of view have been proposed, but researchers have generally studied either the involvement of firms in a community or the integration of FLOSS into their market strategy, but not both. In this article, we argue for a more structured and global analysis, based on the tools of industrial economics, and thus starting from the basic conditions of the computer market and of the buyers' competence in software development (the "dominant user's skill"). This conceptual framework helps to distinguish the different types of corporate behavior we see in the FLOSS ecosystem and more specifically their varying degrees of involvement.

'Free'/'libre' or 'open source' software, industrial economics, dominant user's skill, specificity of the assets.

Résumé

Ces dernières années, la diffusion du logiciel libre, ou open source, représente une des évolutions les plus importantes de l'industrie des technologies de l'information. Dans un contexte d'une économie basée sur la connaissance, ce modèle apparaît comme exemplaire pour de nombreuses industries, où la quantité de connaissance qu'il faut maîtriser est trop grande pour être maîtrisée par un seul agent, même puissant. Considérer la connaissance comme une ressource partagée implique de repenser le concept de chaîne de valeur, car la richesse est générée par les usages de cette base de connaissance (services, produits complémentaires) et non plus de la connaissance par elle-même. Si l'on se place dans une perspective d'économie industrielle "classique", cette restructuration de la valeur doit être étudiée au niveau de l'écosystème global (qui produit quoi entre les entreprises et les universités, entre les utilisateurs et les producteurs, etc.), mais aussi au niveau industriel (une fois que le rôle de l'industrie est compris, comment celle-ci s'organise). De nombreuses explications ont été proposées, mais, la plupart du temps, les chercheurs étudient soit l'implication des entreprises dans les communautés, soit l'intégration du logiciel libre dans leurs stratégies commerciales, rarement les deux. Dans cet article, nous défendons l'idée d'une approche plus structurée et globale, partant des conditions initiales du marché de l'informatique et des compétences des acheteurs en terme de développement logiciel (les compétences de l'utilisateur "représentatif"). Ce cadre conceptuel permet d'éclairer les différents comportements des entreprises que l'on constate dans l'écosystème libre, et spécifiquement la variation de leur implication.

Logiciel libre, économie industrielle, compétence de l'utilisateur représentatif, spécificité des actifs.

JEL: L11, L15, L22, L86

1 LUSSI, M@rsouin (Institut Telecom Bretagne & UEB), Nicolas.Jullien@telecom-bretagne.eu and CNRS / GREQAM and IDEP (CNRS), Jean-Benoit.Zimmermann@univmed.fr

1 Introduction.

"Free"/"libre" or "open source" software (FLOSS) is software whose source-code, which is the explicit expression of the programming work, remains openly accessible. Until recently, it was considered that FLOSS only concerned programmers interested in building and sharing a base of programs developed for their own needs (Lakhani and von Hippel, 2003; Lakhani and Wolf, 2005; Demazière et al., 2006). Today, open source software is increasingly integrated into many commercial offers (e.g., Novell buying Ximian and SuSE, Sun open-sourcing its operating system, IBM open sourcing its development tool software Eclipse, and even Microsoft, who recently decided to distribute some of its software products under open license[2]). Iansiti and Richards (2006) identified, amongst the various FLOSS projects, a "money-driven cluster" where "IT vendors' motives are economic. In this cluster, significant investments have been made in projects that will serve as complementary assets to drive revenues to vendors' core businesses". Lakhani and Wolf (2005), analyzing the results of an investigation of 684 software developers involved in 287 FLOSS projects, found that "a majority of [their] respondents are skilled and experienced professionals working in IT-related jobs, with approximately 40 percent being paid to participate in the FLOSS project".

This paradoxical situation, in which commercial business relies on the existence and durability of non-market activities, is a challenge to industrial economic theory. It clearly has something to do with issues of "coopetition" (Brandenburger and Nalebuff, 1996). As in any cooperative agreement devoted to technology or knowledge development, agents pool assets together in a "pre-competitive" phase and share the fruit of their efforts before returning to competition (Crémer et al., 1990; Bhattacharya and Guriev, 2006). A FLOSS project, on the contrary, is an open game in which the list of players is not bounded ex-ante by a cooperative agreement and the product is a public good that cannot be privately appropriated by the players. This corresponds more to the formation of a consortium for the production of a standard[3].

FLOSS can be considered as an extreme case of "open innovation" (Chesbrough, 2003), defined as "a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology" (Chesbrough, 2006). In this paradigm, the question is to understand which part of intellectual property players may have to open up and which part they must control to build their business (Harison and Koski, 2010; Henkel, 2006).

In each period of the history of computers, certain players have become dominant by controlling some specific assets while others were opened up: with the 360 series, in the 1960s, IBM controlled the computer, but allowed a degree of freedom in the design of independent software (and software producers); with the PC, Microsoft and Intel controlled (and still control) the operating system and a key hardware component, the microprocessor, but the design of the machine was opened up and allowed competition in that part of the market. Can FLOSS be considered as a new form of industrial organization for the computer industry? If so, which asset(s) should FLOSS-based computer firms control?

Industrial economic theory (Shepherd, 1990) explains that an industry is characterized by the basic conditions of each kind of activity: characteristics of the products, of the users - hence of the demand -, but also of the legal environment (intellectual property protection, for instance). These basic conditions shape the main aspects of the market structure (source of added value, competitive advantages, barriers to entry) and the nature of the competition (firms' behavior in terms of price, position, etc.) The efficiency of the firms (their performance) depends on their strategy (behavior, organization) being well-adapted to the market structure, and on their capacity to reshape this market structure - by increasing the barriers to entry, for instance (Tirole, 1989).

More precisely, however, this has to do with Teece's theory of technological innovation and which part of "specialized assets", more or less dependent on the innovation, a firm must control to succeed on the market (Teece, 1986). In fact, we argue that FLOSS corresponds to the emergence in the computer industry of the problem of managing what Teece et al. (1997) called "dynamic capabilities", i.e., the continuous evolution of demand and innovation[4].

2 http://solutions.journaldunet.com/0404/040407_microsoft.shtml

3 What we mean is that a player offers a standard by developing software that the other players can adopt and help to develop. This "unilateral" adoption is usually called 'bandwagon' in the literature on standards (see for instance Farrell and Saloner, 1985).

4 Dynamic capability is defined as "the firm's ability to integrate, build and reconfigure internal and

So in this article, we propose a global analysis of the computer industry and its evolution to explain the emergence of FLOSS as a form of industrial organization, before looking at firms' business to identify the particular asset firms sell in a FLOSS environment.

The article is organized as follows: in section 2 we look at FLOSS on a "macro" industrial level, to determine what, in the history of the computer industry and its evolution, can explain the diffusion of FLOSS. In section 3 we discuss the place of FLOSS as a source of competitive advantage and we introduce the role of the users. In section 4 we discuss of the variety of involvements of firms in FLOSS and how these level and mode of involvement can be explained by the type of users likely to be found in each sub-sector of the IT industry. Then we conclude on perspectives for open innovation regimes.

2 The evolution of the computer industry.

2.1 Characteristics of the evolution.

The evolution of the computer industry since its emergence in the middle of the last century has been studied by Genthon (1995), Dréan (1996), Langlois and Mowery (1996), Mowery (1996) and Steinmueller (1996). We can distinguish three main periods, each starting with a new technology that made possible the design of a new offer for new users. But in each case, the owner of the key asset of the technology was dominant. Zimmermann (1995) and Gérard-Varet and Zimmermann (1985) distinguished three stages in the construction of a complex good: the components (called "elementary technologies"), which are used to create "generic products" or platforms, which have to be tuned to meet certain uses (called "characteristics of use"). The passage from one stage to another is a technological act that must be industrialized. The first (from component to product) is called "technologies of architecturing" and the second (from generic product to usable product) is called "technologies of use". The history of the computer industry is the story of the successive emergence of the dominant design and of the industrial organization to produce it for these two "technological acts".

A dominant technological concept for a dominant demand...

In the first period (mid 1940s to mid 1960s), there was no real differentiation between hardware and software, and computers were 'unique'. They were research products, built for a unique project. In this pre-paradigmatic stage, the users were of the "Von Hippel" type (that we denote VH), who may act as "sources of innovation" (von Hippel, 1988, 1986), able to contribute to hardware development by proposing improvements or modifications, developing it by themselves or at least able to design the technical specifications.

In the second period (early 1960s to early 1980s), thanks to technological progress (miniaturization of transistors, compilers and operating systems), the scope of use extended in two directions: a reduction in the size and price of computers, which increased the number of organizations able to afford them, and an increase in computing capacities, allowing the same computer to serve different uses. But the main evolution was initiated by IBM, with the release of the 360 series, the first family of computers sharing the same operating system. This was the first dominant design of the industry. The computer had become a "classical" good, to be changed once no longer efficient or too old, but without losing investments made in software, because as the program evolves, grows in size, or serves a growing number of users, you only have to move to more powerful hardware.

This allowed computers to reach a new category of the demand, by becoming tools for the centralized processing of information for organizations (statistics, payment of salaries, etc.), firms. And over the course of this period, the size of organizations having access to this tool decreased. Users were no longer able to contribute to the hardware, but they developed strong skills in software development.

The third period began in the late 1970s, with the arrival of the microprocessor. The dominant technological concept and design were in the organization of PC production, where the hardware architecture has been made public and open for competition, but with one single operating system and microprocessor. This proved to be the most efficient way to meet demand in terms of both innovation and price: there are less

external competences to address rapidly changing environments" (Teece et al., 1997).

compatibility problems with programs that are designed for one single architecture, but users are no longer tied to one computer producer, and that increases the competition. The second evolution in design is "package programs". Programs were no longer developed for a single user, but the same program could be packaged and distributed to different people or organizations, in the same way as for other tangible goods. What had happened in the previous period in terms of hardware design now happened for software, with the emergence of a dominant design. Once again the scope of use extended in two directions (increase in power and reduction in size and price of low-end computers). The third period is that of personal, firstly professional and now private information processing. Initially dominated by "Kogut-Metiu Users" (KM)[5], who are not able to contribute to software development but who are innovation takers and sensitive to the technical quality of the offer, the market has growingly dominated by "naive users" (N). These latter are not endowed with noticeable technical skills and are only price sensitive [6].

Specialized assets and control of the industry.

Since the computer was at that time a tool for specialists, with each project allowing producers and users to better understand the possibilities of such machines, the first period was dominated by learning by using, with significant R&D costs. The more one participated in projects, the more able one became to propose innovations for the next project, thanks to the knowledge accumulated. This explains the quick emergence of seven dominant firms (in the USA).

In the second period, this learning-by-using effect did not disappear, as users were able to keep their home-made programs while changing their hardware. This possibility also created the dominant increasing return to adoption effect (Arthur, 1989): technological interrelations. As, factually, a program was developed for and worked with one single operating system, it became difficult for a customer to break the commercial relation, once initiated, with a producer. In return, this customer no longer even needed to understand the hardware part of the machine. The second period was initiated by IBM, and at the end of the period, IBM was the dominant firm (even having to face an antitrust case in the US), although newcomers, HP and Digital, had gained significant positions with mini-computers. If the innovation resided in the operating system, the specialized asset of the period was the distribution network, as you needed to convince customers to adopt your technology to develop their programs. Once that had been achieved, technological interrelations meant that these customers would incur substantial costs if they switched to another family run by another operating system. And with more customers, not only could they invest more in R&D to develop the efficiency of their computer family, but they could also spend more on marketing to capture new customers. And the efficiency of the machines was precisely what the second-period dominant user wanted. So once again, this favored a concentration in manufacturing business, even if on different offers.

The interrelation effect has not disappeared in the third period. But it is dominated by economy of scope, principally because of the development of standardized programs, running on few architectures, reducing development costs (on that particular point, see Mowery, 1996). Of course, as in the previous period, the winners in the computer segment were those who controlled the key elements of the computer, central in terms of technological interrelation: operating systems still, but also micro-processors. They were the companies that benefited most from the economies of scale, as competition brought prices down in other sectors, in particular for the machines which had been a source of high profit before, but also for other components. But new winners in software packages emerged, in more or less broad niche markets. SAP, Oracle and Business Objects are classic examples of the successful newcomers of this period. The access to customers and their needs was the co-specialized asset, as computers had been in the previous period. Once the customer has invested in a software technology, he is tied to that technology by investment in learning. And the more customers it has, the more firms can invest in R&D to develop the efficiency or the functionality of their platforms, and the more they can spend on marketing to capture new customers and/or their feedback to improve the product[7].

In a nutshell, what history teaches us is that in the computer industry, technological evolution allows the construction of new offers, new dominant designs, better-suited to meet new demand characteristics. And each time this happens, the users and their feedback are the key co-specialized asset that firms must control

5 In reference to the concept of "frontier-users" proposed by Kogut and Metiu (2001).

6 We use here a typology of users close to the one defined by Gérard-Varet and Zimmermann (1985), distinguishing between so-called naive, sophisticated and designer users, but here our preoccupation is rather oriented to the capability of these users to contribute to software improvement.

7 For a detailed analysis of the strategies of these firms, see Cusumano (2004).

to succeed in promoting offers based on new technologies. This remains true for the computer industry of today.

2.2 The current industrial organization.

Hardware.

When speaking of computers, we think about machines that are more or less dedicated to specific uses. At one extreme, computers can be used for a wide scope of applications provided by 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. In between, mobile devices like PDAs or mobile phones are built to support a growing number of applications[8].

Vertical competitive advantage is given by better performances/cost ratios (for instance cheaper laptops or better computation capacities for servers or high quality laptops), while horizontal differentiation is based on the integration of new features and high performance tools (e.g. Samsung's folding cell phone display), or on market segmentation through hard-soft-content bundling on new features or applications (e.g. "Mario Brothers" video games only being available on Nintendo machines).

But in terms of the purpose of these machines, and thus the skill of the people buying them, the structure of the markets and of the competition varies. We will take the example of the computer market to illustrate this.

1. Servers are intended to manage, deliver and protect information on the networks. They must be high-performance, stable and compatible with network standards. They are bought by VH users. Microcomputers (with a growing market share for laptop computers) are bought by end users, mainly as personal computers. In the server market, several Unix systems still exist, and this is a case of horizontal differentiation as they are not compatible, so users have to choose between them. For high end customers or needs, mainframes still exist with dedicated operating systems. In the case of the open source Unix, some users prefer BSD (free, open or net) to Linux.
   So even if a growing share of the market is supplied by PC servers running either Linux or Microsoft, it is clear that quality, purpose and niche market strategies are possible, because users are able to evaluate the performance and the suitability to their needs, and are ready to pay for that.
2. In the personal computer market, Apple has a marginal market share, as does Linux, and the Windows-Intel couple dominates the market. IBM sold its PC division to Lenovo in 2004, because it was no longer profitable after Compaq cut prices in the mid 1990s, and the difficulties of Dell today[9] prove that the PC market is dominated by a price war. This is not surprising, as the dominant user is naive and thus only price sensitive.
   The consequence is that firms are continuously seeking to reduce their costs and prices, as it is difficult for them to differentiate horizontally.

Software and service.

Today, according to Cusumano (2004, chap. 2), the application market can be divided into service and product, and for the product side into business specialized offers (which we will call "package offers") and global, "platform offers". We will follow his distinction.

Package offers.

The practice of combined offers, or **packages**, integrating a standard base and customized services has made its mark in the field of professional solutions, for company management systems (ERP, whose symbolic model is SAP), IT tools ("middleware" applications, compilers, development tools such as those proposed by the Ilog company), and the solutions specific to a branch or profession (such as the subsequent version of

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

9 http://economictimes.indiatimes.com/Infotech/Hardware/Dell_may_sell_its_plants_worldwide_Reports/articleshow/3449300.cms

computer-aided design proposed by the company Dassault Systems).

The producer sells "three A services" (Jullien and Zimmermann, 2006): quality Assurance, Adaptation (more or less fast) to the user's needs, and Assistance with using the tool. This is the model of "sustained technical capacity" (Delaunay and Gadray, 1992; Gadray, 1996). The core competence of the firm here is to make the product evolve following line with the needs of the users, but to make this evolution "sustainable" (i.e., ensuring the product remains appropriable and bug-free). If these tools are professional, users are skilled enough to express their requirements (for instance, if they are doctors, that the product is up to date regarding drugs and drug interactions). But they are not always skilled enough in computer science to develop these requirements by themselves, or even to translate them into tender specifications. Here, it is the content of the users' feedback that may vary according to their computer skills.

Platform manufacturers.

They are probably the most studied. These software publishers have broadened the scope of their offer either by supplying a variety of application tools that can be combined with their core product or by offering multiple versions of the latter. This enables them to better meet users' specific needs while keeping production costs down. The archetypal example of such a "platform strategy" is Microsoft, which now offers different versions of its operating system for servers, corporate users and private individuals, as does its open-source competitor RedHat. The same kind of strategy is followed by Oracle, which sells professional applications developed on its database technology, and which has recently bought BEA and SUN, after other takeovers, to enlarge its applications portfolio. Another example is provided by Symbian in the field of operating systems (OS) for mobile applications.

In a nutshell, they are involved in a classic arbitration over standards[10]: to attract the maximum number of users to the platform in order to attract the maximum number of application producers, and vice versa. The history of Linux distribution publishers is another example of the importance of user skills in the creation of a market. RedHat, SuSE and Mandriva (formally Mandrakesoft) were among the first commercial players to enter the market using FLOSS. This could be seen as obvious on a mass market with rather naive users and significant price-based competition. But today, the retail store sales of OS packages represent a negligible part of the revenue of such firms[11], and a major share is targeted on the industrial market. This can be explained by the development of broadband connection. But more than that, user skills matter. PCs are shipped with a pre-installed OS, and few buyers are skilled enough to install a different one. And they have little incentive to do so, since the existing OS has already been paid for with the computer. On the emerging OS for PC/server market, things work differently. Most of the users, of VH or KM type, are aware of the technical issues involved in installing and configuring an OS. It is also easier to buy a machine without an operating system installed. FLOSS gives them access to a more open and more adaptable Unix-like operating system than they could find in the traditional Unix offer, and they are able to choose the Unix they prefer. So, even if they are less price sensitive, FLOSS-based servers may help to differentiate vertically (better quality over price ratio) and horizontally (with the existence of niche Unix).

Service companies.

The largest ones (IBM, Cap Gemini) endeavor to develop a global approach to IS and company organization (by acquiring strategic consultancy companies such as Ernst & Young for Cap Gemini), while remaining less dependent on one type of software, so as to be able to adapt to the constraints and to the current circumstances of these customers. But the retail service companies behave in the same way, supplying infrastructure on a smaller, more local scale (maintenance of a single server, instead of a global infrastructure), either at a more specialized level, for example in terms of sector (e.g. maintenance services for the food-processing industry), or on a more reduced software base (distributors-installers-adapters of one of the platforms; these are Microsoft, Oracle, or RedHat "certified" companies). The vocation of all these companies is to develop, in the customer's interest, individualized solutions and to support these solutions.

10 On standard theories, see the discussions by Katz and Shapiro (1985, 1986, 1994), Teece (1986), Langlois and Robertson (1992, 1995), and for a review of literature, West (2003, 2004).

11 RedHat stopped this activity (see financial report 2006, p. 31); the consumer market (including distributors, OEM sales, e-commerce and Club) represented 2.54 million euros (45% of the total earnings) showing a 23.4% decrease for Mandriva in the 2005-2006 fiscal year; SuSE has been bought by Novell, so these revenues are diluted.

We are approaching what Delaunay and Gadray (1992) and Gadray (1996) described as the "provision of human capacities", in the sense that what makes their singularity (or their core competence) is that they bring together a team of specialists not only in different software but also in their customers' activities. In the following, we will call this **"architect strategy"**. In other words, the efficiency of these firms lies in producing tender specifications that meet their clients' needs. If these companies are technical agnostics, in that they have to install the tools their clients need (or want), it is obvious that the greater their mastery of a tool, the easier its adaptation is and the easier their job is. This widens the strategy field, as firms may differentiate vertically (increasing the number of tools mastered or the number of professional domains covered), but also horizontally, specializing in one domain or software, as do SAP consultants. But in any case, once again, the more computer-skilled their clients are, the easier the discussion will be (De Bandt, 1998).

Actually, the Internet has already impacted these specializations, pushing firms to include more services in their offers or even to design new ways of selling software-based applications, such as SaaS (software as a service) (Cusumano, 2004, pp 86-127; Campbell-Kelly and Garcia-Swartz, 2007).

2.3 Internet innovation, a new phase for the industry.

During the 1990s, with the arrival of the Internet, the principal technical evolution in information technology was, of course, the generalization of computer networking, both inside and outside organizations. Miniaturization also led to the appearance of a new range of "nomad" products ("organizers" (Psion and Palm), music players, mobile phones). This is in line with the constant evolution of information technology products. We have gone from a single machine, dedicated to one task known in advance and reserved for the entire organization, to multiple, linked machines which are used to carry out different tasks, varying over time, and which are integrated within various organizations. Networking, exchanging between heterogeneous systems and communication between these machines have all become crucial.

Thus, because of the spread of the Internet and the growth of exchange outside the organization, network externalities have become the most important source of increasing returns to adoption.

Within client firms, the demand has become more and more heterogeneous with the networking of various systems and the need for users working in the firm to share the same tools. Software programs (and more particularly, software packages) have to be adapted to the needs and knowledge of every individual without losing the benefit of economies of scale, in other words the standardization of the programs on which the solution is based. It is then logical that client firms should seek more open solutions, which guarantee them greater control. For example, what the Internet did was not to offer a "protocol" for the simple transmission of data, since this already existed, but to offer a protocol that was simple and flexible enough to impose itself as a standard for exchange.

In parallel with this evolution, software program technologies have also evolved (Horn, 2000b, pp 126-128): the arrival of object programming languages (C++, Java) allowed existing software components to be re-used. This has led to the concept of "modular software programs": the idea is to develop an ensemble of small software programs (modules or software components), which each have a specific function. They can be associated with and used on any machine, since their communication interfaces are standard. What characterizes the technological evolution of software is thus the increasing interdependence between software programs, while the software components that are re-used are becoming increasingly refined and specialized (Zimmermann, 1998). This system can only function if components are indeed re-usable, that is to say, if producers agree on a mechanism to standardize interfaces and to ensure the stability of these standards over time.

This led Horn (2004) to assert that we have entered a new phase in production: "mass custom-made production", increasing the service part of packaged software sales. Judging by the past, this probably heralds an evolution in the business models and structure of the industry. And as already explained (Dang Nguyen and Pénard, 1999; Genthon and Phan, 1999; Jullien, 1999), the spread of FLOSS is consubstantial with the spread of the Internet.

3. FLOSS as the new frontier for the computer industry organization?

FLOSS can be a source of competitive advantage for firms that take part in or lead its development, but the

nature and level of these firms' involvement varies considerably from one market segment to another. Our main argument here is that this variation can be explored by taking into account the characteristics of the consumers addressed in each market segment, and more particularly their level of skill. The more skilled the users are, the easier it is to introduce horizontal differentiation to meet their needs more precisely, thus creating niche markets. Conversely, when users are too computer illiterate, competition is restricted to prices and this limits firms' investment.

Internet tools were developed in universities, and distributed under free licenses (BSD, for the most): NCSA Web server, the Apache ancestor, Sendmail, Bind... At the beginning of the spread of the Internet within organizations (firms, administrations), servers were installed by engineers who had discovered these tools at university, sometimes without any significant budget. They installed what they knew at the lowest cost: FLOSS products. Apache for Web server, PHP or Python as language for dynamic Web pages are still the leading tools in their branch for Internet application.

3.1 Specific advantages for mass custom-made production.

FLOSS has specific advantages regarding the evolution of demand, improving quality and meeting norms.

Software quality.

More than mere public research products, FLOSS programs were, first and foremost, tools developed by user-experts, to meet their own needs. The low quality of closed software packages and, especially, the difficulty of making them evolve was one of the fundamental reasons for Richard Stallman's initiative[12]. These user-experts are behind many libre software development initiatives (including Linux, Apache and Samba) and their improvement. And as far as these flagship software programs are concerned, this form of organization has obtained remarkable results in term of quality and quick improvements[13].

This is undoubtedly due to the free availability of the sources, allowing skilled users to test the software programs, to study their code and to correct it if they find errors. The higher the number of contributors, the greater the chance that one of them will find any error that may exist, and will know how to correct it. But libre programs are also tools (languages) and programming rules that make this reading possible. All this helps to guarantee minimum thresholds of robustness for the software.

Other widely distributed libre programs are program development tools (compilers, such as GCC C/C++ compiler, development environment, such as Emacs or Eclipse). The reasons are threefold: they are tools used by computer professionals who are able and willing to develop or adapt their working tools, they are the first tools you need to develop software programs, and their efficiency is very important for program efficiency. That is why FSF's first products were such programs, and particularly the GCC compiler.

Meeting norms.

Co-operative work and the fact that the software programs are often a collection of simultaneously evolving small-scale projects, also requires that the communication interface should be made public and "normalized"[14].

Open codes make it easier to check this compatibility and, if need be, to modify the software programs. It is also remarkable that, in order to avoid the reproduction of diverging versions of Unix, computer firms have set up organizations to guarantee the compatibility of the various versions and distribution of Linux. They also publish technical recommendations on how to program the applications so that they can work with this

12 Stallman "invented" the concept of FLOSS, with the creation of the GNU/GPL license and of the Free Software Foundation, the organization which produces them; see http://www.fsf.org/gnu/thegnuproject.html. See http://www.gnu.org/prep/standards.html for technical recommendations on how to program GNU software.

13 On the structure of libre development, besides Raymond (1998a, 1998b, 1999), one can also refer to Lakhani and von Hippel (2003). See Tzu-Ying and Jen-Fang (2004) for a survey and an analysis of the efficiency of on-line user communities, Bessen (2002) and Baldwin and Clark (2003) for a theoretical analysis of the impact of libre code architecture on the efficiency of libre development. The latter argue that libre may be seen as a new development "institution" (p. 35 et seq.).

14 In the sense that they respect public formats whose evolution is decided collectively.

system in the same spirit as the POSIX standard[15]. Firms use libre programs as professional tools to collectively coordinate the creation of components and software program bricks which are both reliable and, especially, "normalized". Up to now, this collective, normalized base has been lacking within the information technology industry (Dréan, 1996). This normalization of the components used to build "mass custom-made products" should help to improve the quality of this production, because the services based on them may be of better quality.
Based on the historical evolution of the computer industry, there are convergent signs suggesting that FLOSS is the industrial organization “of the Internet years”, on the condition that firms develop sustainable business, compatible with the way communities work. In the last decade, an abundant and growing literature has discussed this question.

3.2 Floss involvement and the role of users

Since the mid-1990s, a growing number of commercial firm, either new entrants or incumbents, have decided to integrate FLOSS products in their own specific offer or toolboxes, even investing by different means in FLOSS development. Of course these new emerging strategies must be understood in the light of IP protection prevailing in each market segment and the need to strengthen competitive advantages or to rely on new ones.
Regarding the degree of involvement in FLOSS dynamics, the more active actors seems to be found in sectors where software development and use is either a core activity or a crucial condition for performances, as it is the case for server manufacturers or architects of information systems (adoption of Linux by IBM, HP since the beginning of the years 2000). At the other extreme, the weakest involvement is found amongst hardware suppliers that can only feel concerned by FLOSS for compatibility and price purposes.
When FLOSS adoption is related to marginal aspects of differentiation, it seems to have little impact on industrial structure and competition. This is generally the case for most of hardware producers, when hard-soft-content is no longer bundled (servers, computers, Personal Communication Tools, DVD and MP3 players...)
Surprisingly, FLOSS diffusion impacts mainly firms in software based industries. This has to be understood regarding how their core competences have evolved and shifted significantly. Their main challenge is less and less to supply a “software solution” to a given problem at a given time, but increasingly to deal with short to long term uncertainty over IT system production and management. Users ask for solutions able to protect them against uncertainty, granting interoperability, bug resolution, the satisfaction of new needs and the integration of technical advances . The trade-off between available solutions is not posed in terms of their cost of acquisition but of their “TCO” (total cost of ownership), in which the future costs and the costs for granting interoperability and adaptability have to be estimated. This is precisely what architects, business programs and platform producers sell to skilled users, aware of these problems and signals. On these markets the FLOSS organization seems to represent an asset for producers, who can display their involvement and succeed in building sustainable business models (see the examples of RedHat, MySQL or, in France Linagora). But, as explained before, this is only an asset if the market regards FLOSS as providing a value added to the product, i.e. if this brings the users a potential for increasing their utility.
How and why may those different users contribute directly or indirectly to FLOSS projects? First of all, contribution does not necessary imply code development but can take various forms in the product development and improvement. Users have to be considered as valuable “sources of innovation” (Von Hippel), not only for program testing and debugging but also for improving the product usability and performances. People decide to contribute if they get interested by the product, or if they have a problem, in which case they can either report the problem directly or through an intermediary, the supplier for instance, that allows the user to pass from a passive to an active use of the project.
Actually, the users, understood as the persons choosing the solution (thus not always being the “end-users”), are rather different from one market to another, causing the competitive advantage to rely on different features.
Let us distinct three main types of users according to their relation to the product and the technology (Zimmermann 1995, Kogut and Metiu 2001, von Hippel 1988, 1996). The first is the category of “Naïve

15 This is the Free Standards Group (`http://www.freestandards.org/`). Members of this committee include: Red Hat, Mandriva, SuSE/Novell, VA Software, Turbo Linux, IBM, SUN, Dell, etc.

customers or users" (that we denote N) who are not endowed with noticeable technical skills and do not individually weigh very much in economic terms. The second is the category of "Kogut-Metiu Users" (KM)[16] who are not able to contribute to software development but can generate new features or innovations by revealing their own needs. Above all, they 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) who act as "sources of innovation" (Von Hippel, 1988) able to contribute to software development by proposing improvements or modifications, developing it by themselves or at least able to design the technical specifications.

Users play a double role, deriving from both their economic and technical standing. Depending on the market, and especially their bargaining power in it, the users are more or less able to select the (technical) offers. At one extreme, users and contracts in the global service/architects market are related to large structures, with substantial buying capacities and generally endowed with significant technical skills. So they are likely to influence economic and technical choices. At the other extreme low price computers address a mass market where individual users, in their vast majority have little budget and/or few skills. Their influence on market evolution is negligible at an individual level but of global importance in terms of elasticity to prices. But this analysis should be nuanced in the case of intermediation by a "prescriber", who orders and defines the characteristics for a large number of machines, destined for mass distribution by his own means (local government for secondary schools in France[17], education in rural area in developing countries[18], ...) That's the reason why, when speaking about the "user", we mean the person who negotiates or chooses the characteristics of the good, who is not always the end user.

Of course different types of users are co-existing in any given market. But the dispersion of users' skills in the related technology and more particularly in software doesn't follow the same distribution from one segment to another. Even if skilled users are likely to be found in any market, they may represent a too small share to play a significant role in it and catch the interest the concerned firms for their specific demand. Conversely, thanks to the Internet, a handful of very talented users around the world can weigh enough together to develop a FLOSS alternative to private offers and contribute to the emergence of a FLOSS business offer. So, what we denote users' skills appears as a subtle mix between competences and number, from wich could yield a weighted sum of competences.

What seems clear from a rather qualitative analysis, and was formally demonstrated in Jullien and Zimmermann (2009), is that the skill of the users matters for understanding the level of firms' involvement in FLOSS. When users are naive, firms may use FLOSS, but only for price reasons, in the same way as they could use freeware. The more VH the users are, the more complex the strategies involving FLOSS, and the greater firms' involvement and participation. In some cases, when users are VH, firms may even produce FLOSS and lead the community, as do Ada Core Technology for Ada 2005 and MySQL AB for MySQL data bases. But in any case, FLOSS is regarded as open source software. This means that firms use FLOSS for technical reasons (sustainability, flexibility) and for innovative reasons (increasing the speed and quality of feedback).

4. What we can learn from the markets?

As seen before, there is a wide diversity of actors in the industry in terms of both products and size. Successive waves of innovations and company strategies have led to a progressive reshaping of the industry borders and structure. For example, Internet has impacted the software production, pushing firms to integrate more services in their offers, designing new ways of selling software bases applications, such as Saas (software as a service) (Cusumano, 2004, pp 86-127; Campbell-Kelly & Garcia-Swartz, 2007), However, the foundations of the industry have remained unchanged, since those described by Gérard-Varet & Zimmermann (1985), Zimmermann (1995), Steinmueller (1996), and Cusumano (2004): IT products are built by assembling hardware and software units in a given architecture, and these products (isolated or integrated into networks) are used as parts of information systems and solutions. On the basis of such technical organization, it is then possible to distinguish three large types of "vertical specialization": i. component producers, ii. computers and IT devices suppliers, iii. software editors and service companies providing

16 In reference to the notion of « frontier-users » put forward by Kogut and Metiu (2001)

17 With the aim to provide "a computer for each pupil": http://www.ordina13.com/, http://www.ordi35.fr/

18 See, for instance, the competition between Microsoft and Mandriva to supply 17,000 computers in Nigeria. http://www.computerworlduk.com/management/government-law/public-sector/news/index.cfm?newsid=6124

applications.
All these segments are concerned with software production, as even chipset manufacturers have to deal with the operating systems embeded in the machine integrating their component. They provide drivers for these operating systems, and 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 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. So we will not investigate further the strategies towards FLOSS in this segment of the industry.
Remain what is traditionally defined as the hardware part (the machines) and the software part (software and services), with, in between the operating system.

4.1. The hardware.

Hardware is increasingly various, from mainframes to netbooks, and from dedicated devices (personal communication tools, video game or music players) to the "swiss knife machines" which are modern computers.
Looking at these markets from the dominant user skill prism helps to understand the adoption of FLOSS within the industry.

1. In the **servers market**, producers have habitually provided proprietary solutions with proprietary Unix[19]. Here suppliers are dealing with highly-skilled VH clients that can make an essential contribution in the 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 widen the servers market from VH users capable of managing their systems by themselves to KM clients, 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[20] one, but the main actors of the Unix "world" have also rapidly developed their own offers, cutting down the sources of vertical differentiation[21].
2. The segment of netbooks, and **low price computers** (LPC) is a mass market where naïve clients are the driving force behind demand, and competition is overall based on prices. When Asus entered the market with its eee-PC, it used Linux for price reasons, because Microsoft Windows Vista was too costly in terms of resources needed and price to be competitive. Since, considering the success of this market, Microsoft has designed a specific, downgraded version of Windows XP for these computers[22]. It is worth noting that, since the middle of 2007, Dell proposes Ubuntu Linux distribution on ones of its first price laptops[23].
3. Between these two cases there is the **high quality computers** (HQC) market, ie computers for firms or computers used to play games, computers requiring good, up-to-date performances. In that segment, exigent users, or frontier, KM users seems to be dominant. It is worth noting that in this desktop market, the main push in favour of open source, for the time being, is driven by organizations or institutions (which we consider as VH users) that take decisions to equip a large number of end-users. Examples are the French "Assemblée Nationale" (French Congress) that has contracted with a service company to install Linux on all the computers provided to MPs[24], or the initiatives of the Nigerian[25] and Macedonian[26] governments for schools, or in the industry, the French automaker Peugeot[27].
So, today, HQC 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

19 See West (2003) for a full discussion of FLOSS strategies in that sector .

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

21 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.

22Eee-PC has been the "most wanted 2007 Christmas gift", according to the constructor, http://eeepc.asus.com/global/

23http://www.dell.com/content/topics/segtopic.aspx/linux_3x?c=us&cs=19&l=en&s=dhs

24 http://www.zdnetasia.com/news/software/0,39044164,61970345,00.htm

25 http://www.zdnet.co.uk/talkback/0,1000001161,39290511-39001070c-20088736o,00.htm .

26 http://www.linuxtoday.com/infrastructure/2007091902626NWDPPB

27 http://www.informationweek.com/news/management/showArticle.jhtml?articleID=201400082

editors (RedHat, SuSE,...) which may lead to another dependence and to conflict relations with the dominant provider. Nevertheless, a possible future evolution in this sense is likely to arise from the pressure of corporate and VH customers becoming more aware of the potentialities of switching to FLOSS. It is worth noting that the Linux offered by HP is part of the enterprise offers branch[28]. In the near future most of the HQCs will probably switch to debundling their machines from the associated OS, to segment more their offer between VH users with the Linux offer and KM users with Windows.

4. **Dedicate digital devices** represent another intermediate case with less skilled customers (KM+N) and a weak degree of involvement on the part of commercial actors into FLOSS, and mainly for compatibility and absorptive capacity purpose.

At one extreme, in the games consoles segment but also to a lesser extent in the music player market, proprietary formats have introduced, a strong bundle of hardware-software-content and FLOSS products are non-existent. Thanks to the MP3 standard or new existing or emerging open standards like Ogg, new entries are always possible in segments like the music players market, but the main actors, like Apple, remain on a strict proprietary strategy. On the contrary, barriers remain high on the video game players market due to the scarcity of independent games capable of running on Linux, unlike the PS2, Xbox and other proprietary standards games. Moreover, when they exist, such games seem harder to obtain for simple users.

On the contrary, there are lots of FLOSS products for **Personal Communication Tools**, or Mobile Computers[29]. Some are proposed by VH users, other by the constructors: :

- if the leader, Nokia only soldl an Internet tablet based on Linux and a development community[30], there are lots of open-source projects around Symbian (partly owned by Nokia, partly by Sony-Ericsson)[31], mainly dedicated to tools for developing applications (libraries, development tools, etc.) and Samsung proposes the first smart phones based on Linux[32];
- the PDA Operating system editor Palmsource is working on the integration of its product on a Linux kernel on its products[33].

For the same reasons as for PC computers, we hardly see naïve or KM people switch from an installed operating system to a FLOSS one. So constructors will continue to drive the market and decide what they integrate in their offer. Implementing Linux on PCT devices may appear as a good strategy to limit differentiation to the core competences of the manufacturers. Operating systems are not at the heart of the products differentiation which is more based on ergonomic aspects and hardware characteristics. In the absence of a still established *de facto* standard, as it stands in the PC market, Linux is to be considered by PCT suppliers, as it is free of charge and benefits from a community of developer-users capable to develop new features and new products outside any proprietary control. In fact, similarly to the PC market, the challenge is the choice of a platform (Operating System) to build the product. Palm is also good example of a company which after having sold its OS division, is now turning toward Linux.

4.2. The software.

1. In the **software platform market**, the Linux distribution market is another very good illustration of the key role of the demand. Linux publlishers, like RedHat, SuSE, Mandriva (formally Mandrakesoft), have been among the first commercial actors to enter the market using FLOSS. This could appear to be obvious on a mass market with rather naive users and a significant price-based competition. But today, the retail store sales of OS packages represent a negligible part of the revenue of such firms[34], and a major part is

28 http://h71028.www7.hp.com/enterprise/cache/309906-0-0-0-121.html

29 See, for instance, http://tuxmobil.org/ a web site dedicated to Linux and mobile computers.

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

31 In June 2008, Nokia announced to be acquiring the whole share of Symbian and open source it under Eclipse license. See the Symbian foundation Web site: http://www.symbianfoundation.org/

32 http://linuxdevices.com/news/NS2854558742.html

33 Palm and Linux: http://news.com.com/2102-1041_3-6175171.html?tag=st.util.print. The web site dedicated by Palm to open source: http://www.palmsource.com/opensource/.

34 RedHat stopped this activity (see financial report 2006, p. 31); the consumer market (including distributors, OEM sales, e-commerce and Club) represented 2.54M€ (45% of the total earnings) showing a 23.4% decrease for Mandriva

targeted to the business market.

One might explain this fact by the development of broadband connection thanks to ADSL. But we believe a more important explanation lies in the skills of the users and the construction of the offer. Consumers buy computers with an OS already installed and few of them are skilled enough to install a different one. Additionally there are no incentives to do so because the pre-installed OS has already been paid for with the computer. So, the diffusion of FLOSS OS on desktop/laptop PCs depends more on the strategies of constructors, as discussed above, than on direct installation by users. And for VH people wanting to install Linux on their PC, other, more technically oriented distributions exist, like Debian, and there is no need to pay for these distributions, available for download on the Web.

On the emerging OS for PC server market, things work differently. Most of the users, of VH or KM type, are aware of the technical questions involved in installing and configuring an OS. It is also easier to buy a machine without an operating system installed, and the relative price of the OS is lower. FLOSS gives 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 gave FLOSS OS publishers an undeniable competitive advantage, at least until that server constructors started to offer PC servers with Linux.

2. In the **business software market**, the more skilled the users are, in terms of software development skills, and (although this is a lesser driving force) in terms of cexpressing functionality requirements, the more FLOSS concepts and industrial related offers are likely to spread.

It is clear that the use of open source business software, enabling savings on the cost of licenses, offers a price advantage. Moreover, the fact that the customer can evaluate the product without buying a license is also an advantage in terms of dissemination. It may even be compulsory when dominant players already exist on the market (such as the database market where MySQL proposes software products competing against those of Oracle, IBM and Microsoft who represent more than 80 % of the market) or when customers are highly sensitive to price (such as the ERP market which increasingly concerns SMEs and where open-source products like ERP5 or tiny ERP are now available). This strategy also enables the association of a corporate brand with a product, therefore increasing the notoriety of the firm through distribution of the latter. Moreover, on these technical markets, especially when the customers are developers, availability of the code promotes cooperation. The producer approves the contributions, ensures stability of the tool and helps developers to use it. If some individual contributor becomes important (in terms of contribution volume/quality/innovative aspect), s/he may be hired by a producer, with reduced recruitment costs and risks (ACT or MySQL but also some small services companies are using this method). By contributing to innovation, the developers (and possibly companies using the tool), are therefore guaranteed that their needs will be taken into account more quickly and integrated into the product (which is a fundamental factor in reducing costs, according to Von Hippel 1988).

Obviously, capitalizing on existing products is more difficult, even if, as Muselli (2002) explained, with the entire control of the software , a dual license strategy can be set up to sell the program when requested by the customers (because, for example, they want to integrate it in a larger, closed, package). This is what companies like Qt or MySQL offer. But, today, the main source of revenue again comes from services, more precisely what we call the “3A services” (assistance, assurance and adaptation to the use). Otherwise, adaptation services must be significant enough to finance development of the product. Therefore, the objective is to transform a handicap (significant investments) into a commercial advantage, by increasing the business feedback from users and by considering openness as a way to reduce transaction costs and as a signal of quality. Currently, the main evolution for those firms is to switch from a demand pull strategy (functionalities are developed to stimulate/create the demand) to an 'on-demand' development (development when required and paid for or carried out by the users).

This explains why open source business products are developed mainly in “business” software (ERP, computer infrastructure software like compilers), where users ready to pay for configuration, maintenance or assistance services are numerous. But the scope could easily extend to many technical/professional software activities.

3. As far as the services of the “**architects” market** is concerned, as Horn (2004) points out, assembling components requires access to the source codes (problem of compatibility), and their adaptation to different needs (of users and other components). They must be available in the form of open-source software (therefore legally modifiable).

The competitive advantage in using free software, in addition to price, is therefore the ability to offer an assembled set of components with greater interoperability, which should increase the quality of the final

2005-2006 fiscal year; SuSE has been bought by Novell, so these revenues are diluted.

product, on a market where the quality of services is one of the recurrent problems (see De Bandt, 1998). Revenues are generated by assembling and adaptation services, as is the case for any traditional service company.
The only uncertainty about the model concerns the availability of the components: who will develop them and who will maintain them? Moreover, the customers of these companies may already have (proprietary) programs installed that need to be taken into account. In the end, an open source strategy could even be a guarantee of means (maximum use of free software), but not a guarantee of the results (use of only free software), unless the customer requests this, since in this situation, he keeps the last word.
Two kinds of firms use FLOSS today : newcomers who specialize in FLOSS architecture, using FLOSS as vertical (price) and horizontal differentiation asset, and incumbents, such as IBM for its service activities[35]. Traditional service firms like Cap Gemini are more agnostic with regard to the technologies used and the intellectual property regime involved. They will generally follow the customers' demand which depends on their ability to keep up with the development of the project. These customers are most often large organizations, skilled computer users that are receptive to the opportunity to integrate the most advanced software components, developed under open licenses. So they are becoming increasingly involved in FLOSS as the market grows and matures[36].

Table 1 below summarize the main type of users likely to be found in each sub-sector of the IT industry.

Table 1. The dominant user type in each IT sub-sector.

Actors/ products	***Dominant user type***	***Comments***
Components	VH	Component producers supply hardware manufacturers, aware of the quality and quality-price aspects of the components they will use, as well as the effects of brand reputation of these latter as a signal of quality for their own products.
Servers	VH	The clients are computer-literate people, able to express needs in technical terms, to develop software for their own needs, and to innovate by themselves.
High Quality Computers	KM	HQC users are somewhat less computer-literate than server users; they can be characterized as “intensive frontier users”. So the market is looking at a good performance-to-price ratio.
Low Price Computers	N +. KM	LPC is a mass market; users have no particular skills except in the case of intermediation by a “prescriber”.
PCT	N + KM	PCT and players are relatively mass markets, but some advanced users (more in the PCT field and particularly in the PDA market) can play a constructive role in the development of new features.
Players	N	
Platform producers	KM + N	For the OS, as for hardware components, most of the end-users buy a computer with an OS already installed. So the actual users in our sense of the term are computer manufacturers, service companies and sophisticated end-users capable of installing an alternative operating system for their proper use or the use of their customers. On other platforms (database, middleware), the users are also computer manufacturers, service companies and highly-skilled users.
Business solution producers	VH/KM depending on the markets	In the business solutions market, users are professionals. They are able to make a technical evaluation of the product, to carry out trials and tests. This means that people may have skills in the functional domain (what they want, how the software works), and sometimes in the technical one (able to adapt or develop software to meet their own needs, especially in the tools for computer professionals market).
Architects	N (+VH)	Large firms and organizations include very sophisticated users (IT division). SMEs or corporate divisions, at local or sectorial level, are clients of very heterogeneous but rather low IT skills. However, clients may be quite precise in the definition of the services they need, and so in the specification of the application characteristics.

35 As explained by Slatter (1992), one of the main strategies for newcomers in technological markets is technological differentiation. Basing its offer on new FLOSS products can be seen as a way for new service companies to differentiate.

36 In 2005 Gartner forecasted that « 2008, 95 percent of Global 2000 organizations will have formal open-source acquisition and management strategies » (http://www.gartner.com/DisplayDocument?doc_cd=125868)

(from Jullien & Zimmermann 2009).

Empirical observation about firms' involvement in FLOSS development can be summarize as so: in the fields where dominant user's skill is either high or very low, firms have invested into FLOSS. When dominant user's skill is intermediate, the dominant design remains that of the classical proprietary model. More precisely, when dominant user's skill is low, competition is price-based and FLOSS helps to provide a cheap solution. When dominant user's skill is high, competition is on quality, services and scalability, and FLOSS because it is modular, helps to design (so with complementary investments from firms), a better offer. But between this two polar cases, for dominant user rewarding quality for a low-medium price, FLOSS may not be a good alternative to proprietary solutions.
As shown by Jullien & Zimmermann (2011), when dominant user's skill is high, the variation of investment into FLOSS development and communities among firms, can be explained, in the spirit of Teece (1986), Teece et al (1997).

As far as business packages are concerned, the specific asset of the producer lies in its package knowledge and in its capacity to manage the dynamics of evolution. This makes the open sourcing of a software the specific asset of the firm which owns it: on the technology markets where the customers are computing developers, revealing the code facilitates cooperation. The producer organizes the collaboration in a "symbiotic" relationship (using the terms of Dahlander et Magnusson, 2005). Developers (possibly companies using the tool), by providing their own innovations, are thereby assured that their needs will be taken into account more rapidly and integrated into the product, a crucial point to reduce their costs (von Hippel, 1988); from the producer's point of view, this decrease the R&D cost as the users provide him/her with new feature requirements and, more original, implementation; on the other hand, only the one who integrates contributions is capable of verifying and of guaranteeing their correct functioning and to help clients to use it. So, a FLOSS based package model means that the firms which publish the software remain heavily involved in its development in order to control it. As their core competence lies on the management of the software edited, the companies should only invest in the software they edit, and the involvement of salaried developers in other projects should not be encouraged.

As far as architects are concerned, to be able to integrate knowledge and innovation from the open-source communities, they have to develop internally efficient capabilities of absorption, an essential condition to capitalize and internalize the communities' contribution and the users' feedbacks to improve their own product quality. Dahlander et Magnusson (2008) working on the relations between firms and open-source communities show that those firms need "to develop sufficient absorptive capacity to benefit from external developments, not only to identify useful external knowledge, but also to assimilate and apply it". This is what has been called a "commensalistic approach" (Dahlander et Magnusson, 2005). This corresponds to the more general assertion from Cohen et Levinthal (1989, 1990) about the necessity for a firm to make internal efforts of R&D a prerequisite for the absorption of external technology. This reflects a change in the technologies used, thus of the complementary assets these firms need to manage, more than in the core competences. Traditional architect firms are not involve in FLOSS development, as they do not use these technologies. But they may have other processes for monitoring the evolution of the complementary asset, the technologies they use. They may participate in editors' training sessions, or conclude "global alliance" with their key partners, as Cap Gemini does[37].

5 Conclusion. Lessons for open innovation regimes.

The FLOSS movement has sometimes been presented as a canonical model of production for the open innovation paradigm (Chesbrough, 2003), and even for the knowledge society. If so, open development may develop in fields where users are skilled enough to initiate the development of open knowledge and have enough market power to force the traditional producers to shift to an open model. The major risk in this model is of killing the goose that lays the golden eggs, discouraging individual participation through over-control or non-cooperative behavior.

These conditions being respected, the open IP regime can be seen as a very efficient solution to the

37 `http://www.capgemini.com/services-and-solutions/by-industry/retail/alliances/` for classical alliance, and `http://searchsystemschannel.techtarget.com/news/article/0,289142,sid99_gci1261207,00.html` with those done with open source world.

Schumpeterian dilemma, insofar as it permits a wide diffusion of knowledge, while favoring innovation, as producers are encouraged to contribute to the development of the product they use/sell.

This regime could be called the "VH open innovation regime", in reference to Von Hippel's seminal work on users as innovators (von Hippel, 1988). Open initiatives have been launched in many industries, such as biotech, remote sensing and chip design. Their chances of success are usually evaluated in terms of the motivation of the participants and the stability of the "community". Our contribution argues for more economic aspects to be taken into account.

References

Arthur B. (1989). Competing technologies, increasing returns and lock-in by historical events, *The Economic Journal*, 99 (March), 116-131.

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

Bessen J. (2002). Open Source Software: Free Provision of Complex Public Goods. Rapport, Research on Innovation. URL: http://www.researchoninnovation.org/online.htm{\#}oss.

Bhattacharya S. and Guriev S. (2006). Patents vs. trade secrets: Knowledge licensing and spillover. *Journal of the Economic Association*, 4(6): 1112–1147.

Brandenburger A. and Nalebuff B. (1996). *Co-opetition: A Revolution Mindset That Combines Competition and Cooperation... The Game Theory Strategy That's Changing the Game of Business*.

Campbell-Kelly M. and Garcia-Swartz D. D. (2007). From Products to Services: The Software Industry in the Internet Era. *SSRN eLibrary*.

Clément -Fontaine M, Jullien N. and Dalle J.-M. (2002). New economic models, new software industry economy. Technical report, RNTL (French National Network for Software Technologies) project, 202 pages. URL: http://www.marsouin.org/IMG/pdf/fichier_rapporte-3.pdf.

Chesbrough H. (2003). *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press, Boston, MA.

Chesbrough H. (2006). *Open innovation: a new paradigm for understanding industrial innovation*, pages 1–12. Chesbrough et al. (2006).

Cohen W. M. and Levinthal D. A. (1989). Innovation and learning: The two faces of R&D. *Economic Journal*, 99:569–596.

Cohen W. M. and Levinthal D. A. (1990). Absorptive capacity, a new perspective of learning and innovation. *Administrative Science Quarterly*, 35:128–152.

Crémer J., d'Aspremont C., and Gérard-Varet L.-A. (1990). Incentives and the existence of pareto-optimal revelation mechanisms. *Journal of Economic Theory*, 51(2):233–254.

Cusumano M. (2004). *The Business of software: what every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad*. Free Press, New York.

Dahlander L. and Magnusson M. G. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy*, 34:481–493.

Dahlander L. and Magnusson M. G. (2008). How do firms make use of open source communities? *Long Range Planning*, 41 (2008):629–649.

Dang Nguyen G. and Pénard T. (1999). Don et coopération dans internet: une nouvelle organisation économique. *Terminal*, (80/81):95–116. Numéro spécial, : le logiciel libre.

De Bandt J. (1998). Les marchés de services informationnels: quelles garanties pour le client, consommateur ou partenaire? *Revue d'économie industrielle*, 86, 4 trimestre:61–84.

Delaunay J.-C. and Gadray J. (1992). *Services in Economic Thought: Three Centuries of Debate*. Kluwer, Dordrecht.

Demazière D., Horn F., and Jullien N. (2006). How free software developers work: The mobilization of 'distant communities. URL: http://ssrn.com/abstract=1301572.

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

Farrell J. and Saloner G. (1985). Standardisation, compatibility and innovation. *Rand Journal of*

Economics, 16:70–83.

GADRAY J. (1996). *L'économie des services*. coll. Repères, La Découverte, Paris. réédition de 1992.

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

GENTHON C. and PHAN D. (1999). Les logiciels libres: un nouveau modèle? *Terminal*, 80/81:167–188. Numéro spécial, *Le logiciel libre*.

GÉRARD-VARET L.-A. and ZIMMERMANN J.-B. (1985). Concept de produit informatique et comportement des agents de l'industrie. In *colloque "Structures économiques et économétrie"*, Lyon.

HARISON E. AND KOSKI H. (2010), Applying Open Innovation in Business Strategies: Evidence from Finnish Software Firms, *Research Policy*, 39, 351-359.

HENKEL J. (2006). The jukebox mode of innovation - a model of commercial open source development. *DRUID Working Paper*, (06-25).

HORN F. (2000b). *L'économie du logiciel. Tome 1: De l'économie de l'informatique à l'économie du logiciel. Tome 2: De l'économie du logiciel à la socio-économie des "ondes de production" des logiciels*. Thèse de doctorat, Université de Lille I, mention: économie industrielle, 570 pages.

HORN F. (2004). *L'économie des logiciels*. repères, la Découverte.

IANSITI M. and RICHARDS G. L. (2006). The business of free software: Enterprise incentives, investment, and motivation in the open source community. *Harvard Business School Working Paper Serie*, (07-028).

JULLIEN N. (1999). Linux: la convergence du monde Unix et du monde PC. *Terminal*, 80/81:43–70. Numéro spécial, *Le logiciel libre*.

JULLIEN N. and ZIMMERMANN J.-B. (2006). New approaches to intellectual property: from open software to knowledge based industrial activities. In LABORY S. and BIANCHI P., editors, *Industrial Handbook on Industrial Policy*, pages 243–264. Edward Elgar (EE). ISBN 978 1 84376 836 4.

JULLIEN N. and ZIMMERMANN J.-B. (2009). Firms' contribution to open-source software and the dominant user's skill. *European Management Revue*, 6:130 – 139.

JULLIEN N. and ZIMMERMANN J.-B. (2011). FLOSS Firms, Users and Communities : A Viable Match? *Journal of Innovation Economics*, N°7, 2011-1, 31-53

KATZ M. L. and SHAPIRO C. (1985). Network externalities, competition, and compatibility. *American Economic Review*, 75:3:424–440.

KATZ M. L. and SHAPIRO C. (1986). Technology adoption in the presence of network externalities. *Journal of Political Economy*, 75/4.

KATZ M. L. and SHAPIRO C. (1994, spring). Systems competition and network effects. *Journal of Economic Perspectives*, 8(4): 93–115.

KOGUT B. M. and METIU A. (2001). Open source software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2): 248–264.

LAKHANI K. and VON HIPPEL E. (2003). How open source software works: Free user to user assistance. *Research Policy*, 32:923–943. URL: `http://opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf`.

LAKHANI K. and WOLF R. (2005). *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*, pages 3–22. Feller et al. (2005).

LANGLOIS R. N. and MOWERY D. C. (1996). The federal government role in the development of the U.S. software industry. In MOWERY D. C., editor, *The International Computer Software Industry, A comparative Study of Industry Evolution and Structure*, pages 53–85. Oxford University Press.

LANGLOIS R. N. and ROBERTSON P. L. (1992). Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries. *Research Policy*, 21(4):297–313.

LANGLOIS R. N. and ROBERTSON P. L. (1995). *Firms, Markets and Economic Change, A Dynamic Theory for Business Institutions*. Routledge, London, New York.

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

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

SHEPHERD W. G. (1990). *The Economics of Industrial Organization*. Prentice Hall International Editions, London. 3rd edition.

SLATTER S. (1992). *Gambling on Growth: How to Manage the Small High-Tech Firm*. Wiley, New York.

STEINMUELLER W. E. (1996). The U.S. Software Industry: An Analysis and Interpretive History. In MOWERY D., editor, *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*. Oxford University Press, Oxford and New-York.

TEECE D. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15 (7): 285–305.

TEECE D., PISANO G., and SHUEN A. (1997). Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7): 509–533.

TIROLE J. (1989). *The Theory of Industrial Organization*. MIT Press.

TZU-YING C. and JEN-FANG L. (2004). A comparative study of online user communities' involvement in product innovation and development. National Cheng Chi University of Technology and Innovation Management, Taiwan, 29 pages. URL: `http://opensource.mit.edu/papers/chanlee.pdf`.

VON HIPPEL E. (1986). Lead users: a source of novel product concepts. *Management Science*, 32(7).

VON HIPPEL E. (1988). *The Sources of Innovation*. Oxford University Press, New York.

WEST J. (2003). How open is open enough? melding proprietary and open source platform strategies. *Research Policy*, 32 (7):1259–1285.

WEST J. (2004). The role of standards in the creation and use of information systems. Seattle, Washington. Standard Making: A Critical Research Frontier for Information Systems workshop.

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

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