Open Source Software: the Role of Nonprofits in Federating Business and Innovation Ecosystems

François LETELLIER
fl@flet.fr

January 2008

ABSTRACT

Due to its function as a key enabling technology for the information society and to its compliance with open standards, infrastructure software is quickly commoditized. Its market therefore evolves to natural monopolies faster than markets for tangible goods. The success of free/libre/open-source software, originally created for ethical reasons, may be explained by interpreting it as a new paradigm which provides effective answers to the structural flaws of the market. Now that mainstream industry players realize how much economical sense this approach makes, they investigate new business and innovation models.

In this context, business-neutral meta-organizations federating vendors, customers and governmental agencies shall target the sustainable development of business ecosystems where stakeholders, widely spread across various geographic and cultural environments, develop beneficial strategies in line with their business and societal requirements. Real world experience from new generation F/L/OSS communities confirm this vision and suggest some balanced principles for their management and governance.

Keywords: free/libre/open-source software, business ecosystems, collective strategies, collaborative engineering, open innovation, intellectual property

1. INTRODUCTION

Future software infrastructures promise to be orders of magnitude more complex than today's, so as to address requirements such as interoperability, security, dependability, usability, testability, reliability, safety, flexibility, accountability – and ubiquity. Not only are these issues complex by nature, but vast consensuses over technology choices will have to be reached so to answer them in a socially acceptable manner. Collaboration will be necessary to address complexity; multilateral thinking will be required to reach consensuses. Building tomorrow's software infrastructures will be a multiplayer game on the global scale.

Collaborative engineering based on the "hacker attitude" and free access to source code have proved extremely effective in building quality software and in building consensus. However, not all companies today seem comfortable with free/libre[1]/open-source (FLOSS) communities, for a full spectrum of reasons ranging from cultural differences to legal stumbling stones. In this paper, we explore the rationale for new generation organizations which aim at bridging the gap between FLOSS software communities and the business world while keeping best practices of both.

This paper is organized as follows: in section 2 we present some characteristics of infrastructure software and explain why such technologies tend to be quickly commoditized. Section 3 presents theoretical and observational justifications that, due to commoditization, the market for infrastructure software suffers from structural flaws that leave vendors and users unsatisfied. Section 4 proposes to reconsider the virtues of the FLOSS process on the basis of its economical effectiveness and capacity to address market flaws with pragmatism. Section 5 argues that FLOSS business models can contribute to building a robust business and innovation ecosystem where vendors and customers can find their place. Section 6 concludes.

2. COMMODITIZATION OF INFRASTRUCTURE SOFTWARE

2.1. Software as an Information Good

Software is a pure information artifact, distinct from hardware layers and from the delivery of services. As a kind of information good (or *knowledge* goods) it is *aspatial*, *nonrival* and *discrete* by nature: *its use by one agent does not degrade its usefulness to a yet different agent; its extent is not localized to a physical spatial neighborhood; when instantiated, [it is] created to some fixed, discrete quantity usually taken to be 1, as there is then one copy of the item*. [2] Infrastructure software enables and facilitates the development of complex distributed systems designed to process information. *Operating systems* and *middleware* typically fall into this category. Not targeted to the desktop and virtually invisible to the final user, infrastructure software is often described as hidden or "buried". Nevertheless, as computing becomes increasingly pervasive and computer systems increasingly complex, infrastructure software appears as a key enabling technology for the development of the information society in the digital age.

2.2. Open Standards and Interoperability

Infrastructure software hides the physical heterogeneity of hardware platforms and, to some extent, the distributed nature of computation. In addition to the functional features (such as communication capabilities), it provides uniform high-level programming interfaces which facilitate software engineering. In the current state of our technology, the above objectives are met by designing shared *encodings*, *formats*, *protocols*, *interfaces* and akin kinds of specifications - often referred to as standards.

De jure standards are defined by independent standardization bodies, such as public authorities or gatherings of industry stakeholders. *De facto* standards are initially defined

[1] In 2000, the European Commission introduced the term *libre software* to avoid the ambiguity of the English word "free". In this paper, we use the expression Free/Libre/Open-Source Software (FLOSS) to speak of software distributed under a license that allows complete access to the source code and grants use, modification and redistribution rights.

by an isolated industry player or by a small party and later on vastly adopted by other actors. From a competitive viewpoint, *de facto* standards are typically promoted by a market leader, or a forerunner in an emerging market, and later on used by this leader to retain its market domination.

Independently of their *de jure* or *de facto* status, *open standards* are characterized by the following properties:

- They are explicitly documented. For this reason, they enable the development of interchangeable software
- They are openly accessible and free to implement by everyone. This enables free competition and cooperation between technology vendors
- They are usually defined to accommodate both providers and consumers needs, in a consensual process involving a panel of potential vendors and potential users

Open standards play a pivotal role in the development of software infrastructures in the information society: *two features are essential to the deployment of the information infrastructure needed by the information society: one is a seamless interconnection of networks and the other that the services and applications which build on them should be able to work together (interoperability)* [9].

2.3. Commoditization

A *commodity* is a good used as a building block for many different purposes, sourced by more than one producer and defined by uniform quality standards [13]. High tech commodities are building blocks for more complex systems: computer parts, RAM, microchips are sourced by many different manufacturers, comply with very strict specifications and are used by many different assemblers.

Strict compliance with standards tends to make solutions interchangeable [52]. *Making software a commodity by developing an industry of reusable components was set as a goal in the early days of software engineering. While significant progress has been made, this still remains a long term challenge* [1]. Whenever implementation of these standards is free, as allowed by open standards, several providers are likely to develop standard-compliant products. The very existence of an open standard promises that a significant market exists for such products.

For these reasons, open-standard-compliant infrastructure software has the characteristics of a commodity: several providers, many uses, and interchangeability.

3. STRUCTURAL FLAWS OF THE INFRASTRCUTURE SOFTWARE MARKET

3.1. Free Market, Perfect Competition

Perfect competition does not exist in the real world – and the meaning of the word *competition* is subtly different in traditional economics and in the neoclassical theory. Neoclassicals rule out the assumptions of homogeneity across products and of perfect information. They see competition as a dynamic process aiming to gain advantages through product differentiation. The ultimate goal for sellers is to monopolize the market – and to eliminate competitors.

Still, the market for open-standard-compliant infrastructure software in the digital age is fairly close to the traditional vision of perfect competition (at this point, we must make it clear that we are here speaking of the market for software licenses – not intermediary markets for intellectual property rights ownership – see 3.3). Publishing software is a straightforward and painless process. Although complex and powerful, software only requires little investment in terms of production tools. Its distribution is easy and inexpensive; a worldwide outreach can be targeted at virtually no cost through the Internet, which at the same time ensures efficient and free information.

3.2. Competition in Commodity Markets

Competition in quantity is an unrealistic model for information goods such as software that can be duplicated *ad infinitum* at no cost. In model of *competition in price* [30], each competitor chooses its own output price, while assuming the other competitors' price constant. Products are homogeneous across all competitors, who may deliver any quantity as required to serve the customers. Information flows freely and perfectly between sellers and buyers.

In such model, a Nash equilibrium [31] is reached when no competitor makes any profit. Many economists believe that this result (the "Bertrand paradox") is descriptive of real, highly competitive markets even though the conclusion that no competitor makes any profit when the market reaches equilibrium sounds against common sense. Authors have proposed solutions to the paradox by introducing capacity constraints (Edgeworth) or assuming differences between products (other than price). None of these solutions seem to apply well to standard-compliant software: there's no limit in the sellers' production capacity, and standard compliance significantly reduces differentiation between products.

3.3. Unlimited Supply and Drastic Economies of Scale

On the infrastructure software market, a *product* is not any specific computer program, it is actually an instance (or copy) of this program. According to the laws on intellectual property rights (IPRs), the proprietor of the copyright grants a right to use (the exact definition of use being variable) the software [46]. Getting a copy of the software is a prerequisite to benefit from the right to use it. In this paper, we don't cover the issues related to unlawful uses of software. For this reason, we assume that any transfer of the right to use a computer program is accompanied with the necessary transmission of a copy of the software. Conversely, we assume that any transmission of a copy of the software is accompanied with the transfer of some rights to use it for some purposes, under some conditions. For simplicity we call *license* the bundle of a copy of software with the right to use it.

Software can be duplicated *ad infinitum* with no loss of quality. Duplication costs are extremely low, actually negligible when compared to development costs. The production of licenses is therefore not limited due to any technical constraint. A licenses shortage may only originate in a vendor's unwillingness to provide them.

3.4. Very Low Prices

Due to drastic economies of scale on software licenses, whenever the market is big enough, the Bertrand-Nash equilibrium is for a quantity close to market saturation and to a very low price. For a worldwide software market of millions of users, the theoretical equilibrium is for a price millions times smaller than the software development cost (first unit). This prediction based on the rather simplistic model is confirmed by more sophisticated models (e.g. Arrow-Debreu): *because of zero marginal costs of reproduction, the present value of an*

intellectual asset would, under perfect competition, turn out to be zero [49].

Observations corroborate theoretical predictions: literally hundred thousands programs are available *gratis* on the Internet. We do not only speak here of "free software", but also of *freeware* (and to some extent of shareware too) and of "lite" versions of commercial software. Such software often is of very high quality and rivals with commercial offers: GNU/Linux, OpenOffice, PostgreSQL, Internet Explorer, Acrobat Reader are popular examples. Today, businesses can actually be lawfully operated using only gratis software, and the biggest part of the Internet infrastructure relies on free software [52].

The issue of very low software price at first only sounds like a problem from the vendor's perspective. However, low prices come with side effects, including low perceived value and high economical risk for the vendors. In the long run, low prices, and subsequent limited margins, incite those vendors who primarily derive revenues from licensing to put on the market only good-enough products, hence sacrificing software quality, long term support and actual innovation.

3.5. Natural Monopolies

Industries with strong economies of scale are known for their proneness to *natural monopolies*. Once in a dominant position, a vendor may keep the price low enough to deter potential competitors and as high as possible to maximize its profit. In the software industry, anti-competitive moves are specially easy to perform, by cutting down the price when necessary, as anyway the marginal production cost is virtually zero [51].

The tendency of the market's price to decrease quickly due to almost perfect price competition tends to reduce the perceived value of the good. As happens with other information goods, *positive per-item prices are inefficient because they discourage consumption with value greater than marginal cost. Further, very low per item prices will not recover first-copy costs and thus firms will not have an incentive to create new content* [39].

3.6. Oversupply of Information Goods

The winner-takes-all effect is a powerful, even though deceptive, incentive to produce information products, even when financial profit is not the main motivation (notoriety, peer recognition, etc. are common alternate rewards [43]). Even individuals with very limited material resources can ambition to be in the position of serving a worldwide market, at no cost, if their work is compelling enough. At the individual level, success is uncertain; at the global level, the bottom-line is an oversupply of information products. As an example of information goods where oversupply is documented, the literature mentions new classical musical compositions, written in greater numbers than can be performed.

The software oversupply is well exemplified by the thriving of FLOSS projects (hundreds of thousands on SourceForge.org only), and of free/sharewares.

3.7. An Unefficient Market

Unlike free markets for tangible goods, the market for open-standard compliant infrastructure software appears plagued by a vicious circle. Commoditization is a key enabler of adoption, but economies of scale induce a rapid fall of market prices, while creating a radical winner-takes-all effect and consolidating natural monopolies. The perspective of being the winner is a huge incentive for creating new software, hence inducing oversupply.

Even though some companies may perform well during a given period of time, the horse race effect in new markets and the price fall in older markets make business models centered on the commercial licensing of open-standard compliant infrastructure software a predictable failure for the vast majority of competitors. The bottom line is an ill-functioning market which leaves a vast majority of vendors and customers unsatisfied.

4. OPEN SOURCE AS A PARADIGM SHIFT

4.1. Ethical Motivations for Openness

In the early days of computing, access to the source code of any piece of software was the common rule, as software was most often provided as an adjuvant to hardware. This tendency started to change as computers got more widely adopted in the early 80's, as mass storage and networks enabled easier distribution of data and software, and as commodity computers hit the market.

Nevertheless, ever since then, the Free Software Foundation and proponents of *free software* advocated for ethical principles [34] in software production, namely the *computer users' rights to use, study, copy, modify, and redistribute computer programs* [15]. In this vision, access to the source code is little more than a technical means to achieve the above stated goals.

Over time, the movement of *open-source software* appeared with the more pragmatic rationale to set software quality as a main goal, to promote access to the source code as the preferred means to reach this goal and to explicitly target commercial use of open-source software. As stated by the Open-Source Initiative [33]: "*The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. […] We in the open source community have learned that this rapid evolutionary process produces better software than the traditional closed model, in which only a very few programmers can see the source and everybody else must blindly use an opaque block of bits. Open Source Initiative exists to make this case to the commercial world*".

4.2. FLOSS Thriving: an "Exaptation"?

Extensive intellectual property rights are frequently considered as a way to overcome market inefficiency for information goods, for the benefit of innovators and customers.

Without entering the debate about software patenting, it is worth noticing that software oversupply, commoditization and availability of *gratis* software products are a reality even in the countries where software can be patented (e.g.: the USA).

In regions where software is patentable[2], adverse effects of software patents on innovation and their use for anti-competitive purposes have been documented [48]. Some recent studies suggest that for information goods, *markets function optimally and IPRs are either unnecessary or, if they affect allocations, harmful to social efficiency. Creativity and innovation are properly priced in competitive equilibrium, and socially efficient outcomes obtain without the contrivance of IPRs* [2], [49], [50].

[2]at the time of writing, software as such is outside the scope of patents in the European Union

From early programming languages to modern component-oriented methods, software engineering is based on modular design. Modularity makes it (fairly) easy to split a product between a part that is protected by strong IPRs (patents) and a part classically protected by copyright. Commoditization occurs on this second part. Commodity, standard compliant components are good enough for a majority of uses, while strongly protected enhancements create a *de facto* separate market. This interpretation is consistent with Christensen's *conservation of modularity* [36].

In a context where an over protective extension of software IP fails to provide appropriate answers to structural market flaws, the FLOSS approach may have spread and gained momentum because it makes economical sense for infrastructure software development. To use evolutionary jargon, the rapid expansion of FLOSS in the ICT sector [11] may be seen as a successful *exaptation* of *free software*, i.e. *the utilization of a structure or feature for a function other than that for which it was developed [through natural selection]* [35]. Originally imagined to protect the user's freedom, free/open source licensing introduced a mutation in copyright practices which eventually proved economically sound.

This is corroborated by the observation that top motives for adoption of FLOSS by companies are *independence from pricing and licensing policies of big software companies* and *technical superiority* [21] over proprietary software [17], [18].

4.3. Business Models

Because users ask for FLOSS, vendors learn to adapt their offer so to answer the demand. New business models centered on FLOSS appeared shortly after free and later on open-source software flourished: as stated by A. J. Slywotzky, v*alue migration is the shifting of value creating forces. Value migrates from outmoded business models to business designs that are better able to satisfy customers' priorities* [37].

At any given point in time, not the whole software stack is commoditized. One reason is that there are missing, or not-yet-defined, standards. Another reason is that some very specific developments are only relevant to niche markets. The full extent of this latter case is when tailor-made software is specifically developed for one single customer. At the boundary between commoditized and not yet commoditized parts of the software stack lie opportunities to develop successful, yet transient, business models: according to C. Christensen, *when attractive profits disappear at one stage in the value chain because a product becomes modular and commoditized, the opportunity to earn attractive profits with proprietary products will usually emerge at an adjacent stage*. [36]

B. Perens lists four paradigms for software development: *retail; in-house & contract; efforts at collaboration without open source licensing; and open source* [16]. This classification emphasizes the fact that collaboration is a long known production paradigm, but also that the open-source process goes a step further thus becoming a distinct paradigm.

J. Koenig [4] proposes seven business models briefly presented below:

- *Patronage*
  Patronage is a strategy to use FLOSS to proactively foster the commoditization of a given part of the software stack. Motivations may be various: accelerating the adoption of a standard; using open-source as a channel to transfer research results to the industry; balancing the domination of the leader in a monopoly market; undercutting entrenched competition. Revenue generation is not the main goal.
- *Optimization*
  The core of this strategy is to leverage cost savings achieved on commodity to sell added-value, fine-tuned proprietary bricks for specific uses. This strategy is a direct application of the law of conservation of attractive profits.
- *Dual licensing*
  Dual-licensing is a hybrid licensing scheme mixing open-source and close-source options and targeting direct license revenue. A software product is licensed under a commercial license (that grants rights that may include access to the source code and rights of modification/redistribution). Another version of the same product, typically coming with fewer features, is made available to the community under an open-source license. In this scheme, open-source is mainly used as a promotional tactics. As the vendor of the commercial version needs to detain IP rights over the software, contributors to the open-source version need to agree on assigning their copyright to the vendor, possibly in a non exclusive manner.
- *Consulting*
  This option is in no way specific to FLOSS. As the FLOSS process is mainly technology-driven, consultancies offer complementary professional services that bridge the gap between the companies' business expectations and the FLOSS communities: training, architecture, certification, support, customization, fine tuning, etc.
- *Subscription*
  The subscription model intends to derive recurrent revenues from packaging or bundling open-source software along with recurrent services: selection of best-of-breed technologies, integration, maintenance, updates, support, etc. Software oversupply is a good argument in favour of such offers, as complexity of the IT world becomes overwhelming to non-IT companies [40].
- *Hosted*
  The hosted strategy consists in offering software as a service(servitization of a product). This can be done in a very straightforward way by application service providers or in a more indirect way through *infoware* [13], including all kinds of e-business applications. A key competitive advantage is derived from the access to the source code, that enables extreme agility to applications that constantly evolve to meet and anticipate customers needs.
- *Embedded*
  This model directly derives value from the right to sell commodity software. It consists in embedding FLOSS either in hardware products or in more complex software products: in the first case, the use of commodity hardware is likely to be a prerequisite, as commodity FLOSS is more likely to be available and stable on widely adopted hardware. Access to the source code enables the embedded provider to adapt and fine tune FLOSS to the specificities of the target hardware/software architecture.

It is worth noticing that at the time of writing, the taxonomy of FLOSS business model is far from being stabilized in the literature. *Support seller*, *loss leader*, *widget frosting*, *accessorizing*, *service enabler*, *sell it/free it*, *brand licensing*, *community enabler* is another classification which is

widely used. This diversity demonstrates that even though a common misconception is that "there is no business model for open-source", there are actually many ways of doing business with FLOSS, and vendors creativity has not yet reached its limits.

Apart from the above models, more classical business models such as *industrialization of service*, *cutting out the middleman*, *loyalty* or *network marketing* may also be applied to FLOSS. The classical *bait and hook* strategy also can be adapted to FLOSS in a very straightforward way: a vendor uses FLOSS to disseminate a technology and sells complements (either proprietary software or service) over which it retains a competitive advantage. To successfully run this model, the vendor needs to avoid open-standards if selling proprietary software complements, or needs to retain a control over IPRs and/or the key developers of the FLOSS, so to claim an unrivalled level of expertise if selling service. In this model, as in dual licensing, source openness is little more than a marketing strategy, although it may help increase software quality.

The unidimensional customer-supplier relationship appears outmoded in the world of FLOSS where various business models play a complementary role throughout the software supply and demand chains. Pure FLOSS players most often follow business models that are a blend of two or more of the models described above, and their success is highly dependant on their ability to gain a keystone position in the ecosystem. In all cases, the choice of the licensing scheme is highly dependant on the business model.

4.4. Windows of Opportunities

As decribed in the theory of adoption expressed by the Chasm group [38], adoption of a new high-tech technology (disruptive innovation) follows a lifecycle roughly segmented in three phases, separated by two "cracks". The first crack (the "chasm") divides the early market (innovators and early adopters) from the mainstream market. Making a new technology cross this crack is key to its success and long lasting adoption.

The second crack divides the mainstream market in two parts: early majority on one side, late majority (conservatives) and laggards on the other. We emphasize on this second crack because it typically separate two groups of customers according to the level of integration in the new technology: *conservatives like to buy preassembled packages, with everything bundled, at a heavily discounted price* [38]. In the specific case of software, serving these expectations from conservatives requires the availability of *low cost*, *commodity* components that can be packaged as turnkey bundles.

We propose to characterize the three phases as follows:

- *Emerging technology*: immature standards; little adoption; interest from the early market
- *Being commoditized*: maturing standards; adoption by the early majority in the mainstream market; fast evolution of the market state towards low-cost, highly adopted offers
- *Fully commoditized*: critical mass of users; generalization in the conservative part of the mainstream market; market state close to the theoretical equilibrium for fully commoditized information goods; transfer of attractive profits to other parts of the stack

We propose three patterns of opportunities for the various business models described above:

- *Transient opportunity*: patronage; optimization; dual licensing; bait and hook
  These business models are typically adversely affected by a wide adoption of the technology. Patronage is pointless once a technology reaches the state of a key enabling technology. Both optimization and dual-licensing bear the risk of finding big competitors or very low cost alternates, most likely in open-source, in their way. This risk is very well exemplified by the emergence of totally free, open-source R/DBMS such as PostgreSQL and Derby that start undercutting Oracle (optimization [4]) and MySQL (dual license [4]) respectively. Bait and hook is likely to fail in the long run, for two different reasons depending on whether the FLOSS bait complies to open standards. If it does, the scheme may fail for the same reasons as dual licensing. If it does not, open standards are likely to prevail in the long run.
- *Early opportunity*: consulting; subscription
  The curve of opportunities is relatively flat. Business opportunities increase as the technology becomes more widely adopted. Lack of skills and support are the two main reasons why companies reject open-source today [22]. Service companies specialized in FLOSS (consulting) and "distros" (subscription) typically turn these lacks into business opportunities. Contributing to the code typically is a competitive advantage for these business models.
- *Late opportunity*: hosted; embedded
  The successful development of these business models depends on technology maturity, i.e. its stability, low operating costs, wide adoption by customers and/or wide availability on commodity hardware. Web hosting providers (hosted) leverage the popularity of the Apache web server and of various scripting languages to propose low-cost web presence. The cost-effective use of GNU/Linux in appliances such as routers (embedded) is made possible by its stability and its availability on commodity processors.

4.5. Usage Models

Open-source business models apply to entities acting on the supply side of the software landscape. We propose another classification, this time suited to users, of four levels of involvement in the development of FLOSS:

- *Reuse*
  FLOSS software is used as cost-effective and/or high-quality alternative to proprietary, close-source software. Access to the source code is anecdotic and is not a major motive for choosing FLOSS [18]. Although this model is the most passive, it contributes to the overall sustainability of the FLOSS projects through adoption and direct network effects.
- *Double-sourcing*
  This option is a variant of *reuse*. FLOSS is used as a partial substitute for proprietary software. Non mission-critical applications are typically migrated to FLOSS solutions with lightweight supporting services; or migration of non critical systems pave the way to

gradual migration of more ccritical parts of the IT infrastructure. Alternatively, scalability of information systems is achieved by complementing a proprietary core architecture with satellite FLOSS bricks. This is a way to lower risks, as the user is no longer dependant of a single technology/provider. This is also a powerful way to negotiate with proprietary vendors.

- *Percolation*
  The idea is there to consider the FLOSS community as an partner in the development and maintenance of software developed in house. The motive is here to mitigate the burden to maintain or enhance software [45]. Code donation is typically done to well-established open-source projects. Code development may be outsourced to a service company under the condition that software be released in open-source. Percolation can be applied to outdated technologies, by software companies willing to remain on the bleeding-edge of technology and to challenge competitors while getting rid of what's no longer a competitive advantage to them [5], or as an entry strategy in an optimization business model: the lower end of a proprietary suite is open-sourced to drive adoption of the remaining proprietary part through indirect network effects.
- *Shared R&D*
  Collaborative development is a paradigm shift from the traditional supplier customer relation that turns tables and put the responsibility for software development on the users themselves. In this model, there is simply no supplier or, better said, the community of collaborating producers of software is the supplier for each one of them. This approach is often compared to communes, *commons* or coops, and should be evaluated taking into account that *about half of internal developments don't work out* [5].

5. THIRD GENERATION OPEN SOURCE ORGANIZATIONS

5.1. Three Generations of Open Source Organizations

Open-source communities address technological issues in a very efficient fashion. However, they leave companies virtually alone when it comes to complementing software with all it takes to make a product: positioning, packaging, customization, training, services, communication, quality assurance, certification of compliance with standards, etc.

Historically, free software first emerged from the efforts of individuals following a form of "hacker ethics". Over time, the informal communities working on free/open source projects felt the necessity to incept legal entities, made of individual members. The Apache Software Foundation is very typical of this second generation of FLOSS organizations: *the membership of the ASF is composed of individuals, not companies* [32].

Now that FLOSS reaches the mainstream, the software industry is ready for a third generation of FLOSS organizations: gatherings of legal entities. The Eclipse foundation, the MMBase foundation, the ObjectWeb Consortium and the OW2 association, the Open-Source Development Lab (OSDL) appear to be forerunners of this new generation. They differ from organizations of an older generation in either of two ways:

- unlike standardization bodies, they target code development instead of standards creation
- unlike FLOSS communities of individuals, they directly and openly involve companies in a business oriented fashion

Consequently, the profile of developers found in third-generation organizations is significantly different from that of the second generation. The typical FLOSS developer was pictured [12] as a technology-enthusiastic, male bright kid in his thirties working on his spare time for personal motives. Communities members increasingly depart from this archetype, with more involvement of professionals of all ages and both genders appointed to contribute to projects on their working hours, according to a corporate rationale.

5.2. Collective Strategies

Business models and usage models are patterns that link economical and technical aspects of FLOSS. The conjunction of several patterns involving various industry players dramatically increases the sustainability of the projects, hence that of the related business models.

Coopetition [24] appears to be a major modality of high tech companies' activity, which encompasses three notions: complementarity between organizations; multiplicity of roles played by a single organization; and governance rules that enable to keep balance between competition and collective strategies. Collective strategies often rely on formal relations, embodied in contracts.

The FLOSS licenses provide a formal framework for the technical part of collaboration: *open source is partnership with rules* [5]. However, when collective strategies come to involve business-oriented legal entities, this legal framework needs to be extended so to address non-technical aspects of collaboration.

Meta-organizations such as cooperatives [44], unions, consortia, federations may provide a legal framework suitable to formalize collective strategies, ensure proper governance and promote coopetition between players from different backgrounds, competitors included [26].

In closed structures such as coops where members share efforts and only members share results [44], collaborative development of domain-specific software makes sense. However, for reasons exposed above, infrastructure software *offers far more value when shared than when used in isolation* [3]. In this context, the openness of FLOSS licenses is fundamental in improving software quality, fighting the "not-invented-here" syndrome [5], distributing technology for the common good and eventually recruiting new members.

5.3. Proactively Building an Ecosystem

Although collaborative development is pivotal in the third generation of FLOSS organizations, the potential of inter-organizational collaboration goes beyond collective strategies [23] to encompass the development of a full *business ecosystem* [25], [42].

A business ecosystem is an economic community supported by a foundation of interacting organizations and individuals [25]. It is typically led by one or a small number of entities. In a non-coercive structure, the leader draws its legitimacy from shared values and beliefs, and from the broadness and accuracy of its vision.

In the FLOSS world, a non-profit meta-organization federating industry stakeholders is in a good position to play this role of a leader. This is neither incompatible with nor prejudicial to its members positioning. *The crucial battle is not*

between individual firms but between networks of firms. Innovations and operations have become a collective activity [41].

Although a meta-organization may play a leading role in the development of a business ecosystem, the ecosystem extends beyond the community of members. License openness plays an instrumental role in expanding the fuzzy boundaries of the ecosystem.

A healthy ecosystem is a scale-free network of industry players characterised by its productivity, robustness and efficiency in creating new niches [41]. These two last points are valuable to customers, because when considered as a whole the ecosystem is free of single points of failure. They are also valuable to members of the ecosystem, because they ensure that new business opportunities keep appearing.

In a healthy business ecosystem, a proportionally small number of players are *keystones*, i.e. entities highly interconnected with other entities in the ecosystem. *Firms following keystone strategies are often small players by obvious measures, and have no presence at all in most niches in their ecosystem. Their influence is exerted not by size, but by the relationships that make them essential to the overall health of the system* [41].

A strong commitment to a meta-organization may be a good way to position a firm as a keystone of the business ecosystem. Agile keystone firms are in a privileged position to tap on new niche opportunities or, in other words, provide computing, software and/or services *on demand*.

5.4. From Value Proposition to Governance

The business model of meta-organizations such as coops, consortia, etc, is *collective*. Leveraging collaboration to reach various goals (e.g. an increased negotiation power or economies of scale in production) is the core of their value proposition. Networks effects [20] also apply to a meta-organization focused on infrastructure software [3].

The *collective* facet brings together vendors that leverage collaboration for the part of their business model which is compatible with collaborative engineering. To some extent, the organization plays the role of an innovation intermediary, fostering *open innovation* [7] in its membership.

The *network effects* facet targets vendors and users altogether, for the part of their usage model which benefits from a wider technology adoption Network effects only make FLOSS superior to proprietary flavours of collaborative engineering for technologies with a significant potential for adoption. Domain specific developments (e.g. those done by the Avalanche cooperative) do not fall into this category.

Because not all business models benefit from collaborative engineering to the same extent, the code base shepherd by a FLOSS meta-organization is fragmented. Some projects are led by companies running business models where leadership is pivotal. Other are collaboratively developed for percolation, shared R&D or patronage purposes. Although the number of contributors to single projects may be low in average, the code base grows as an assembly of complementary components and is finally be the result of a large scale collaboration. The dynamics of *commons-based peer production* [19] is here well exemplified – in coherence with the FLOSS golden rule of modular design.

As a whole, the meta-organization runs a patronage strategy. Its overall value proposition increases with the number of members: over its time of operation, the size of the ObjectWeb consortium increased exponentially (doubling every twelve months), which is a hint that the recruitment pattern was enabled by network effects.

To the users, its value proposition derives from the growth of a business ecosystem of technology suppliers, networks effects and eventually better durability of software.

To the vendors, its value proposition is: a collective strategy with opportunities to become keystones of the ecosystem, an overall patronage strategy compatible with the members' business models, and the use of network effects to proactively increase software adoption.

Such meta-organization needs to remain neutral with regard to their members interests, which means, in the business world, that it is most often positioned as not-for-profit and transparent.

It faces the challenge of becoming a partner to all, and a competitor to none, of its members. The organization cannot be positioned as a software vendor without bearing the risk of entering in direct competition with commercial ISVs in its membership.

An option is to clearly identify a core *platform* distributed by the meta-organization as pure software commodity. The Eclipse Foundation chose this option: *Eclipse is an open source community whose projects are focused on building an open development platform* [10]. Once a technology (typically, an open standard) is selected for the core platform, the meta-organization becomes a competitor to all of its members, or potential members, who promote an alternative option. The risk in this case is that the prevalent influence of one or a few members over the organizations become detrimental to the rest of the community. Another difficulty comes from the licensing scheme: only business models compatible with the core platform license remain accessible to the members.

Another option is to make a clear distinction between each member's offering, in terms of products and services, and that of the meta-organization. While members are ICT players, the organization is positioned as an innovation catalyst, the place where collaboration happens. The benefit to the end users is the increased likeliness that whole products that fit their needs be collectively brought to the market by the ecosystem members [6].

5.5. Societal Motives as a Retrospective Effect

The success of FLOSS brings externalities that may have huge societal consequences. Apart from opportunities for cost containment in public expenditures and intrinsic quality of public FLOSS-based information systems, the transition between an IPR-based to a service-based software economy [43] is an opportunity for local economical development.

In sensitive contexts such as defence, unimpeded and unconditional access to the source code of critical software is a direct advantage of FLOSS over proprietary software. Such technological independence is regarded as increasingly valuable as computing becomes pervasive in the information society. *The idea that the extension of the sphere of influence of the market impinges on societal choices, and that it should not be allowed to develop blindly or in a uniform manner, is now being expressed more clearly both among political leaders, especially in the developing countries, and within civil society* [28].

Public software infrastructures have two good reasons to be considered a (global) public good. The first one lies in their status of public infrastructures *per se*. The second is twofold in itself, and comes from the software nature of these infrastructures. Software, considered in the source code form, is little more that the expression of algorithms, i.e. ways to solve problems, in a structured language akin to mathematical

notation. Some authors argue that for this reason, software should be regarded as the formalized expression of some knowledge, i.e. of a public good [27]. In addition, the source code of any public infrastructure software also is the description of the way this infrastructure works, and therefore, of possibly hidden or unwanted regulations [29]. The possibility of citizen scrutiny of public software infrastructures used for e-government, and more generally, for all publicly founded services, appears a must to ensure democratic control over information society.

It is worth noticing that the *tragedy of the commons* [8], i.e. the abuse of public infrastructure to serve private interests to a point that jeopardizes the very existence of this infrastructure, should be reconsidered in the case of software infrastructures. Because it is aspatial, nonrival by nature and distributable ad infinitum, infrastructure software can be used to any extent without suffering any damage.

Ethical and societal motives for adopting FLOSS in public, defence and e-government software infrastructures become increasingly prevalent in many countries around the world. Governmental action plays a critical role in the success of FLOSS business models through direct economical incentive and through exemplarity which reinforces ethical motivations.

6. CONCLUSION

In this paper, we presented theoretical and empirical evidences that the market for infrastructure software is structurally flawed. We proposed to consider the success of free/libre/open-source software as an evidence that the FLOSS paradigm is an efficient way to cope with structural market flaws. This vision underlies an emerging rationale for the production of open-standards compliant infrastructure software.

Finding an economically efficient way to produce such software may be critical in the Information Society. The rationale we analysed leverages open-standards and the open-source collaborative process to foster the development of a business ecosystem where each player could define its own innovation strategy while contributing to the global sustainability of FLOSS development.

As this vision is significantly different from the classical vendor-supplier relationship, we reviewed business models and usage models for FLOSS infrastructure software. We argued that, altogether, vendors and users, from industry, academia and government, may contribute to common projects and may consolidate a business ecosystem exempt of single points of failure. We briefly argued that this approach appears effective in meeting the user requirements while complying with some governmental and societal expectations.

Traditional FLOSS communities have so far been successful in delivering technically superior software. But the industry is now asking for a new generation of meta-organizations that would help structure and rationalize FLOSS investments by better taking business expectations into account. Their success will depend on their capacity to define and enforce governance principles to ensure fair coopetition between corporate members without losing the creativity of individuals.

BIBLIOGRAPHY

[1] Definition of middleware by the ObjectWeb Consortium, http://middleware.objectweb.org
[2] Quah, Danny (April 2002) “Matching demand and supply in a weightless economy: Market-driven creativity with and without IPRs”
[3] Carr, Nicholas – “IT Doesn’t Matter” in Harvard Business Review
[4] Koenig, John (May 14, 2004) IT Manager’s Journal
[5] Perens, Bruce – http://reviews.infoworld.com/article/04/12/03/49FEopensourceinterview_1.html
[6] CALIBRE – Co-ordination Action for Libre Software Engineering for Open Development Platforms for Software and Services – Deliverable 3.2.
[7] Chesbrough H. (2006) “Open Business Models”
[8] Lloyd, W.F. (1833) “Two Lectures on the Checks to Population” Oxford University Press
[9] “Europe and the global information society”, Bangemann report recommendations to the European Council, (May 26, 1994) – http://europa.eu.int/ISPO/infosoc/backg/bangeman.html
[10] www.eclipse.org
[11] Study on the economic impact of open source software on innovation and the competitiveness of the ICT sector in the EU, November 20, 2006 – UNU-MERIT. http://www.ntu.edu.sg/nbs/sabre/working_papers/24-96.pdf
[12] Lakhani K., Wolf B., Bates J. Boston Consulting Group (2002) “BCG Hacker Survey”
[13] “Tim O’Reilly in a Nutshell” (2004) – O’Reilly
[14] Wall Street Journal (February 15, 1996)
[15] http://www.fsf.org (March 2005)
[16] Perens, Bruce “Economic Paradigms of Software Development” – http://perens.com/Articles/Economic.html
[17] “Free/Libre Open Source Software: Survey and Study” – http://www.berlecon.de/studien/downloads/200207FLOSS_Use.pdf
[18] http://reviews.infoworld.com/article/04/12/03/49FEopensourcesurvey_1.html
[19] Benkler , Yochai (2002) “Coase's Penguin, or Linux and the Nature of the Firm”
[20] http://www.dtc.umn.edu/~odlyzko/doc/metcalfe.pdf
[21] “Linux: Fewer Bugs than Rivals” (December 14, 2004) in Wired - http://www.wired.com/news/linux/0,1411,66022,00.html
[22] “The Cost and Risks of Open Source” (April 2004) by Forrester Research – http://download.microsoft.com/download/7/d/0/7d059de9-1557-415c-8332-920db6f89e44/FRSTRossCosts0404.pdf
[23] Astley, W.G. and Fombrun, C.J (1983) “Collective strategy: social ecology of organizational environments” in Academy of Management Review
[24] Nalebuff, B. and Brandenburger, A. (1996), “Co-opetition”
[25] Moore J.F. (May-June 1993) “Predators and prey: a new ecology of competition” in Harvard Business Review, May-June 1993, pp. 75-86
[26] Bresser, R.K. and Harl, J.E. (1986) “Collective strategy: vice or virtue?” in Academy of Management Review, pp. 408-427
[27] “Knowledge as a Global Public Good”, the World Bank, http://www.worldbank.org/knowledge/chiefecon/articles/undpk2/w2wtoc.htm
[28] “Global Public Goods”, Directorate-General for Development and International Cooperation, Ministry of Foreign Affairs, February 2002 – ISBN 2-11-092966-9- http://www.diplomatie.gouv.fr/cooperation/dgcid/publications/partenariats/biens_gb/pdf/biens_publ_gb.pdf
[29] Lessig, L. (1999) “Code and other laws of the Cyberspace”
[30] Bertrand, J. (1883) “Theorie Mathematique de la Richesse Sociale” in Journal des Savants, 67, 1883, p. 499-508
[31] Nash, J. F., Jr.(1950) “Equilibrium Points in n-Person Games” in Proceeding of the National Academy of Science U.S.A., 36, 1950, p. 48-49
[32] http://www.apache.org/foundation/faq.html as of January 2008.
[33] http://www.opensource.org
[34] Williams, Sam (2002) “Free as in Freedom, Richard Stallman’s Crusade for Free Software”, O’Reilly, 2002 – ISBN 0-596-00287-4
[35] The American Heritage Dictionary of the English Language: Fourth Edition. 2000.
[36] Christensen, C. (2003) “The Innovator’s Solution: Creating and Sustaining Successful Growth”, Harvard Business School Press, 2003
[37] Slywotzky, A. J. (1996) “Value Migration, How to think several moves ahead of the competition”
[38] Moore, G.A. (1991) “Chrossing the Chasm”
[39] Fay, S.A. and MacKie-Mason, J.K. (2003) “Competition Between Firms that Bundle Information Goods” – http://plaza.ufl.edu/faysa/bundle.pdf
[40] Andriole, S. (2002) “Improving Biz/IT Convergence: An Action Plan” in CIO - http://cioupdate.com/reports/article.php/1468381
[41] Iansiti, Marco and Levien, Roy (2004) “The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability” Harvard Business School Press

[42] Gueguen G., Pellegrin-Boucher E. and Torrès O. (2004) "Des « stratégies collectives » aux « écosystèmes d'affaires » : le secteur des logiciels comme illustration" Atelier de Recherche AIMS "Stratégies collectives : vers de nouvelles formes de concurrence", Mai 2004, Montpellier
[43] Raymond, E.S. (1999) "The Cathedral and the Bazaar"
[44] Bray, Hiawatha (2004) "Group pushes software sharing" Boston.com, July 2004
[45] Bac, Christian; Berger, Olivier; Desbordes, Véronique and Hamet, Benoit "Why and how-to contribute to libre software when you integrate them into an in-house application?" to appear.
[46] Berne Convention for the Protection of Literary and Artistic Works (1886)
[47] UK Prime Minister's speech about the Millenium Bug, March 30, 1998 - http://www.number-10.gov.uk/output/Page1161.asp
[48] "To Promote Innovation: The Proper Balance of Competition and Patent Law and Policy", A Report by the Federal Trade Commission, October 2003 – http://www.ftc.gov/os/2003/10/innovationrpt.pdf
[49] Quah, D. (May 2002) "24/7 Competitive Innovation" – http://econ.lse.ac.uk/staff/dquah/p/0204-247.pdf
[50] Boldrin, M. and Levine, D.K. (January 2003) "Perfectly Competitive Innovation", January 2003 – http://www.dklevine.com/papers/pci23.pdf
[51] Fuller, T. (March 15, 2003) "For Microsoft, market dominance doesn't seem enough" in The International Herald Tribune
[52] "Your Open Source Plan" in CIO, March 15, 2003 - http://www.cio.com/archive/031503/opensource.html