'LIBRE' SOFTWARE: TURNING FADS INTO INSTITUTIONS?

Jean-Michel Dalle
Sorbonne Université
jean-michel.dalle@sorbonne-universite.fr

Nicolas Jullien
Telecom Bretagne & ICI
Technopole de Brest Iroise - F-29285 Brest Cedex
Tel: (33) (0)298001245 - Fax: (33) (0)298001173
e-mail: < Nicolas.Jullien@telecom-bretagne.eu>

This version, September 2001: forthcoming, Research Policy.

ACKOWLEDGEMENTS: The authors wish to thank Godefroy Dang Nguyen, Laurent Kott, Thierry Pénard, Morris Teubal, Jean-Benoît Zimmermann who provided helpful comments on earlier versions of this article. Michel Callon and an anonymous referee also significantly helped to shape this version.

ABSTRACT: The article presents an economic analysis of Libre software and of its sustainability as a new economic model for software. We underline the role of Libre software development communities and analyze incentives of both kernel and obscure developers. We emphasize the role of the so-called 'public' licenses to provide an appropriate institutional framework. We show that several features of Libre software also allow it to improve faster than proprietary software, and therefore to achieve strong market performance when competing against existing standards, even when proprietary software producers react. We illustrate our point using a simple local and global interaction model to study the technological competition between Linux and Windows on the server operating system market. We finally argue that Libre software could turn from a fad into an efficient economic institution for instance to correct inefficiencies due to network externalities, if sufficient initial momentum could be created through public intervention, then without the help of feelings such as Microsoft-phobia.

KEY WORDS: Libre software, Linux, Community, Incentives, Network Effects, Network Externalities, Technological Competition.

1. 'Libre' software

What should we call software distributed with its sources *and* with the right to modify and redistribute it as long as it retains the same characteristics, e.g. software like Linux, the emerging new star in the operating systems (OS) market? Such software is often referred to as 'open-source' software or as 'free' software, the first expression being now somewhat more frequent than the second. Unfortunately, both expressions are misleading and somewhat inadequate. First, Linux is not necessarily free, and is certainly not going to be free for most users, as distributions of Linux are now being *sold* at classical retail stores by various companies (Red Hat, Correll, Süse, etc.) while many other companies are selling costly *services* to Linux users for them to truly benefit from their new OS. Second, there are many examples of open-source software which are indeed *proprietary* and which certainly does not qualify as being the kind of software which we are considering, since openness of sources does not guarantee that it can be modified by anyone, and certainly not that anyone is allowed to 'freely' redistribute it. Linux is neither 'free' software nor 'open-source' software: following a recent report to the European Commission, we would like to suggest calling Linux and its fellows 'Libre' software (with a capital L to avoid misunderstanding). in French, 'Libre' refers to liberty and to freedom and not to gratuity[1]. Libre software is software distributed with its sources *and* with the right to modify it and to redistribute it as soon as it remains Libre.

So Linux is a piece of Libre software, and Libre software is gradually appearing as one of the most fashionable and possibly one of the most interesting new economic models in today's software industry. It indeed allows users to co-operate – essentially through the Internet – by making most of the time marginal improvements to a given piece of software before it is redistributed once modified. In this way, each user rapidly benefits from innovations brought by all others. Libre software is thus a very seductive concept, but all the interest it has attracted has not only been theoretical since it has also much to do with pure commercial success. Linux, the most well-known example of Libre software, has clearly appeared as a major challenge to Microsoft OS's for servers (NT and now 2000): it is reported to run more than 25% of Internet servers[2] and its commercial shipments are seen by some analysts to be growing faster than any other operating software (25% vs. 10% per year[3]). Meanwhile, the web server "Apache" is leading its market with no less than a 60% market share[4] -, and the mail server "Sendmail" unambiguously dominates its own market, not forgetting many other examples.

Such success is challenging for economists, and specially so since Libre software was truly born more as a fad –rooted in anti-proprietary software and more specifically in anti-Microsoft and anti-Windows attitudes – than as a sustainable

1 It would certainly not be the first time that either English or French borrows words from each other's language when they are clearly more appropriate.

2 http://leb.net/hzo/ioscount/

3 http://www.idc.com/Data/software/content/SW033199PR.htm

4 http://www.netcraft.co.uk/Survey/

economic model or as a relevant economic institution. Here comes then the major economic question this article is attempting to address: will Libre software be able to turn from a fad into an economic model and an economic institution? We believe the answer is twofold.

First, we have to understand how Libre software 'works', which notably refers to the incentives issue (section 2). It is indeed extremely puzzling for classical economic theory that developers choose to participate in Libre software development whereas they seemingly get no reward from doing so. In this context, we argue here that several incentives structures co-exist and apply differently to *heterogeneous* developers, namely, to *kernel* developers or to *obscure* developers. But incentives are only part of the Libre software issue, and might indeed be the *simplest* part of the economics of open-source and Libre software. We argue here that Libre software depends upon a *community* of heterogeneous developers with its associated *institutions*: specially, no appropriate incentives would exist without a credible commitment framework as it is provided by *"public" licenses*, like the General Public License (GPL), that guarantee that Libre software protected by GPL will always remain Libre while it is gradually diffused and improved. According to us, Libre software points at a *new economic and institutional model for creative (epistemic) communities*, which should be compared both with patents and with the conventions of Open Science in the scientific community.

Second, Libre software not only 'works', but also works well, and we do not either have any satisfactory explanation yet for Libre software market performance (section 3), although such performance is both critical for the sustainability of the associated economic model and extremely puzzling for an economist, as Lerner and Tirole (2000) themselves recognize[5]. We argue here that Libre software improves *faster* than proprietary software due to several characteristic factors of the Libre software development model, namely higher increasing returns associated with adoptions by users who are also innovators, and a more efficient redistribution of these returns back to users through faster and more frequent release strategies. Taking as an example the battle between Linux and Windows on the market for server operating systems, we show with a simple stochastic interaction competition model that this 'faster improvement' property is a good candidate to explain why Libre software is able to defeat dominant proprietary standards, even when proprietary software competitors react quickly and strongly, conditional on sufficient *initial momentum* provided for instance by proselytic behaviors.

Both aspects of the Libre software economic model are finally relevant to question the sustainability of the Libre software development model. To be sustainable, Libre software clearly needs creative communities with both kernel and obscure developers. But to be sustainable, Libre software also needs initial momentum. If Libre software was proved to improve collective welfare, for instance by counterbalancing monopoly trends in markets dominated by

5 "Aspects of the future of open source development process, however, remain somewhat difficult to predict with 'off-the-shelf' economic models." (Lerner & Tirole, 2000, p. 1).

network effects, then some kind of public intervention might be relevant to help create Libre software communities and to recreate initial momentum by means other than pure cultural feelings. To sum up, we conclude that Libre software might very well be turning into an efficient institution, and that further research is needed, and obviously fairly soon, to determine where and when – for which kind of software, notably – such an institution would be efficient and should be encouraged, and when it should not.

2. The economics of Libre software[6] communities

2.1 Incentives: simple explanations

Why do Libre software developers disclose proprietary information i.e. why do they contribute for free to Libre software development? A simple explanation lies in reputation effects and associated expected gains, or else in signaling effects and in career concerns (Dalle & Jullien, 2000; Lerner & Tirole, 2000). Both explanations are indeed similar and essentially build upon the existence of what we have called 'ancillary business companies', i.e. companies providing services to Libre software users, be they individuals or organizations, which are not directly supported by the development community (like commercial packaging, after-sale support or simply maintenance and customization of installed systems, etc.).

These companies are now very common: at least a few of them exist for almost all major Libre software, most of which have been created by or in association with *kernel* Libre software developers (such as Linus Torvalds and Alan Cox for the Red Hat company). They very regularly offer job and even equity to kernel developers and therefore contribute to creating rational expectations about the ability of kernel developers to transform into profit their reputation acquired while contributing to Libre software development: lines of code are « signed » by their authors, and knowledge of who did what is widely accessible in the development community.

Although such incentives are targeted at kernel developers, they also partly apply to *obscure* developers. Contributing to obscure aspects of software

---

6 We limit our inquiries here to Libre *software*, although several attempts – largely unsuccessful for now – have been made to apply a Libre model to other types of goods. The codified nature of software (Cowan & Foray, 1997) clearly is a key element in this respect as it allows for actual cumulative work, a key condition of the Libre model. Another point has to do with the fact that software has extremely low production costs, contrary to almost all other economic goods which on the contrary imply significant investments to be produced. These investments would be out of reach of any Libre community, and would probably be also rejected by business firms since the Libre model also implies that they would not be protected by intellectual property in doing so, therefore implying weak barriers to entry. Although further studies are needed here, we conjecture that the Libre model has only limited potential applications outside of the software industry.

development can indeed open the way for subsequent kernel developments, either in the same or in another Libre software community, while contributions, be they small, directly represent a positive signal for many job applications. Kernel and obscure developers belong to the same *epistemic* community: they share a common language and a common goal, which is not in the first place to improve their own skills but essentially to contribute to a collective piece of work (Cohendet & al., 2000). However, associated profit expectations are to be relatively weak for obscure developers as there is clearly a relatively poor reputation associated with the programming of hundreds of lines of relatively uninteresting code.

But for obscure developers, a weak expected reward associated with disclosure is counterbalanced by low creation costs, since we are considering *innovative users* (Von Hippel, 1988) who develop obscure pieces of code for their own interest, and also by a very low reward associated with appropriation (non-disclosure) as obscure work has indeed almost no 'tradable' value. Furthermore, similar solutions are also certainly to be proposed by other members of the community, generally making them, or at least some of them, better off if they disclose first[7]. Think for instance of a developer who happens to use a rather rare printer: considering that he is able to develop the associated Linux driver, or to adapt an existing driver whose source code is in fact open and available, why should he *not* disclose his work? Considering further that they are numerous such developers, he or she is nearly sure that at least one of them will contribute to that particular piece of Libre software development. And there are, indeed, many different such contributions in major Libre software communities among which the best is selected (hence improving efficiency further).

2.2 Incentives: less simple explanations

These two explanations, simple as they are, also rely on the existence of *credible commitment* mechanisms, provided by a key feature of Libre software i.e. the so-called public licenses or GPL (general public licenses), the best known of which is the GNU license[8], generally referred to as the « CopyLeft » protection scheme. Incentives to disclose knowledge would indeed obviously be much weaker if someone was able to appropriate the associated reward, be it through profit or through reputation. All Libre software developers are reasonably sure that their CopyLeft *protected* work is to remain non-proprietary, as CopyLeft means that it is to be freely read, modified and redistributed but it certainly does not allow anyone to use it in any proprietary closed form. GPL licenses guarantee that Libre software is actually not only open-source, but precisely *Libre*.

In fact, economic creativity, being highly subject to positive externalities, generally implies some kind of an associated institutional and/or conventional enforcement mechanism: since knowledge disclosure is a key condition of further knowledge improvement – knowledge is essentially built upon previous

7 See also Harhoff, Henkel & Von Hippel (2000) and Lakhani & Von Hippel (2000) for other related inquiries.

8 http://www.fsf.org/copyleft/. GNU is a « recursive » acronym for GNU's Not Unix.

knowledge –, the social value of creating and disclosing knowledge is significantly higher than its individually appropriable value and there is a need for well-adapted institutional and/or conventional devices, which of course often rely on customized property regimes (Romer, 1993; David & Foray, 1995; Dalle, 2000). This is both the case for patents and for the scientific community, to name but two major examples. Patents as an institutional device correct this by granting patentees – who have to publish their discovery – with temporary monopoly power, with possible enforcement by law courts. In the scientific community, a similar mechanism is provided by conventional means: reputation is granted to scientists who happen to be the first to publish results (Dasgupta & David, 1994). Historically closer to the scientific community (Jullien, 1999), Libre software fundamentally appears as a kind of an *anti-patent* device, as property is not granted but denied. Denying appropriation paradoxically creates incentives towards disclosure as induced spillovers will neither be appropriated by the inventor *nor by others*.

		Patents	*Open-Science*	*Libre software*
Creation and Disclosure	*Incentives*	Monopoly power	Reputation by peer recognition	Reputation, signaling, career concerns and profit
	Enforcement	Law courts via infringement trials	Conventional, self regulation of the research community	CopyLeft and GPL licenses as anti-patent devices[9]
Distribution	*Devices for information exchange and pooling*	Patent databases maintained by public agencies	Scientific journals, Conferences, email	Internet (email, mailing lists, web sites, newsgroups)

Table 1: Different institutional devices dealing with creativity.

Public licenses have proved to be an appropriate tool for transforming communities of practices – of users – into epistemic – creative – communities (Cohendet, Créplet & Dupouët, 2000). As an immediate corollary – and as an indirect proof of what we are stating here –, we are also provided with a straightforward explanation for the difficulties encountered by business firms, such as Netscape, when they try to implement open-source and/or Libre features in their previously proprietary development model. They have mainly been unsuccessful in motivating large and creative enough Libre software communities: even when they hire kernel developers, or manage to be publicly supported by a few of them, they are unable to attract sufficient numbers of other developers, notably obscure ones, as long as they try to cling to not-

9 Although there has been no trial yet, and although some aspects of public licenses remain questionable: see the work of Mélanie Clément-Fontaine at http://crao.net/gpl. One of the major questions has to do with the necessity of public involvement, for instance to secure public licenses by making sure that existing laws properly allow for the existence and sustainability of such licenses. As for now, the CopyLeft scheme has emerged as an alternative institution without any public intervention.

completely-public, and therefore somewhat-still-proprietary, licenses. Many such attempts have aborted due to inappropriate licensing schemes, most of the time due to rational doubts of potential developers about the credibility of the commitment by the applicant firm to a Libre software development model. This is indeed an important lesson for all proprietary software producers interested in fostering Libre software processes associated with some of their products[10].

This last issue is all the more important as hybrid organizations between Libre software communities and business firms is a candidate as an explanation for the ever-increasing efficiency of Libre software. Firms essentially deal with what Libre software communities cannot do – they sort of 'supplement' Libre software communities – while earning money from selling associated services. They notably contribute to the development of more numerous obscure parts of Libre software and therefore considerably improve the user-friendliness of Libre software *products*. They also support a better coordination of Libre software projects: as a matter of fact, although many Libre software communities have been able to implement quite amazing organizational schemes – this is notably the case for the Linux community –, there is still some doubt about their general ability in this respect.

In any case, it should be noted as a conclusion to this section that no reference has been made here to anti-Microsoft feelings or even to purely cultural issues regarding the involvement of developers into Libre software communities. In fact, Linux is just one particular example of Libre software, and recent and older Libre software seems to fit even better into the framework we have outlined here. The Libre software model therefore appears as perfectly sustainable as far as incentives are concerned, and we have now to tackle the other issue, about Libre software market performances against proprietary software, not only because the future economic role of Libre software will also depend on its ability to win competitive situations against proprietary products, but also very simply because incentives to be part of a Libre software community considerably depend on positive expectations about Libre software market performance.

3. An assessment of Libre software efficiency and market performance

3.1 Libre software improves faster

10 As a matter of fact, the ability of would-be ancillary companies to deal with Libre software communities is to be a key condition of their success: they have to deal with a community of « geeks », to quote a word now fashionable even among geeks themselves. This topic was indeed listed among main risk factors in a recent Red Hat IPO prospectus: « Negative reaction within the open source community to our business strategy could harm our reputation and business [···]. [Some] have suggested that [...] we are trying to dominate the market for Linux-based operating systems and the open source community [...]. This type of reaction, if widely [...], could harm our reputation, diminish the Red Hat brand and result in decreased revenue. » Attracting both kernel and obscure developers, with associated dynamic club effects, is indeed a key issue here, and several firms now propose consultancy and technology services to businesses willing to do so.

As Eric S. Raymond[11], a major Libre software advocate, once put it: « From nearly the beginning, [Linux] was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet. Quality was maintained [... by the] strategy of releasing every week and getting feedback from hundreds of users within days. » As we have already outlined, the *efficiency* of Libre software indeed comes from the fact that it is supported by a community which benefits from extremely low development costs and from powerful communication technologies thanks the Internet, from emails to mailing lists and to Web servers. Innovation is thus easily decentralized (Cohen, 1983) and numerous developers are able to push and publish their ideas while only the best are selected, a feature which proprietary software producers have difficulties in coping with. Moreover, Libre software innovators also being users (Von Hippel, 1988), Libre software programs are developed to cope with problems which users really face, thus improving considerably their efficiency through much quicker and better development feed-back. As a comparison, proprietary software producers are instead far removed from their clients, and typically have to organize huge marketing studies to try to decipher their needs.

Still following Eric S. Raymond, the Libre model is not only critical for Libre software improvement, i.e. for bug fixing, it has also other considerable consequences on the way newer versions are *released*: another major difference between Libre and proprietary software. Proprietary software users usually have to pay for newer, although sometimes only slightly improved, versions, and often have to wait for a long time before appropriate patches are released to fix even important bugs. On the contrary, Libre software improvements are regularly and rapidly accessible through the Internet. To summarize, lower development costs and better and quicker feed-back from a community of developers also result in very different *release strategies*. Therefore the pace of improvement of both proprietary and Libre software will be critically different not only because improvements cost less and are more efficient, but also because they are made accessible to users in more efficient ways.

To put it differently, Libre software generally appears as a more efficient way than proprietary software of dealing with positive external economies associated with technological development and diffusion. The result is faster improvement of Libre software as compared to proprietary software. The Libre model allows a more extensive redistribution to consumers of increasing returns associated with technological adoption not only because its diffusion generates higher increasing returns per se, but also because they are not appropriated by producer through opportunistic release strategies. This should remind us of a black box within Arthur's (1989) classical model of technological competition, according to which increasing returns of adoption are sort of assumed to be re-invested in technological improvement by producers, i.e. in a way redistributed to consumers. In fact, proprietary software producers *choose* between profits and investments in further improvements of their products. They select release strategies according to their market power such that they help them secure higher margins and recurrent income, even when this is not satisfactory for

11 http ://www.tuxedo.org/~esr/

consumers[12]. In other words, the rate of improvement of the utility of a technology depends on the extent of its diffusion but is also influenced by choices of producers between alternative release strategies: as for Libre software, it will be structurally higher than for proprietary software.

3.2. Linux vs. Windows

This last aspect is of course of considerable importance for the assessment of Libre software market performance in competitive situations. To further illustrate this point, we now turn to the competition between Linux and Windows in the server operating systems market. Apart from being fashionable, this situation is interesting for at least three main reasons: first of all, it concerns operating systems which are both at the core of any computer and highly subject to network effects associated with compatibility issues; secondly, it is a situation where a new entrant – Linux – is trying to invade a market already dominated by an existing standard – Windows –; and thirdly, Linux is also subject to strong positive local externalities due to the proselytism of Linux adopters: there are for instance numerous associations such as Lugs (Linux Users Groups) who organize regular events to promote its use, while Linux users generally tend to actively promote its use and to publicize its quality and interest to other potential users in their "neighborhoods" i.e. among their acquaintances. As a matter of fact, the ongoing success of Linux – which is continuously and consistently gaining market shares, as we have mentioned above – would be very difficult to assess if we were only to consider global positive externalities, even under the hypothesis that Libre software improves faster than proprietary software, as Arthur and followers[13] have shown.

We therefore turn to a local and global interaction model. We consider a population of heterogeneous potential adopters with different decision rules. Depending on the adopters, quality, performance (frequency of errors), availability and variety of dedicated software, easiness of installation and comfort of use, direct and indirect costs (buying, training, maintenance, upgrades, dedicated software), will indeed be more or less relevant parameters to evaluate the utility of alternative software solutions. To give but one example, such valuations will indeed most certainly be very different from a computer hacker to an unskilled computer user in a major company. This is why we turn to statistical analysis, accounting for the statistical propensity of a randomly chosen given adopter to adopt either one of the two competing technologies.

Yet, individual valuations are also highly sensitive to both local and global externalities. As is now well-known, global characteristics of products, like quality, performance, availability and variety of dedicated software, price, depend more or less directly on their diffusion level. As a consequence, the

12 Not to speak of monopolists' strategies.

13 To put it briefly, Arthur's model neglects local interactions whereas technological adoption generally depends also on the previous choices of a subset of local 'relevant' neighbors (David, 1988). See e.g. Durlauf (1993), David, Dalle & Foray (1998), Dalle (1995, 1997), Kirman (1993) for various modeling exercises using local interactions.

statistical propensity of a random adopter to choose a given technology will also be a function of the market share of this technology and even more so in the case of network technologies for which compatibility issues play a major role. These compatibility and externality issues are also specially prominent within the limited set of other adopters with whom a given adopter often interacts, i.e. *locally*, a specially important feature for operating systems due to regular file exchange among users.

What we therefore have to study is the statistical propensity of potential adopters a given population to adopt each technology conditional on "previous" global and local adoption patterns. We consider here the following statistical demand function[14]:

$$\mathbf{Prob}\left[\textit{Agent A adopts Linux}\right] = \frac{th\left[a\left(\left(x_l - \alpha x_W\right) + b\left(X_l - \beta X_W\right)\right)\right] + 1}{2}$$

where:

• the hyperbolic tangent function (and also adding *1* and dividing by 2) here is just to normalize probabilities of adtoption between –1 and +1;

• x_l (resp. x_w) is the proportion of agent A's neighbors who have adopted Linux (resp. Windows);

• X_l (resp. X_w) is the proportion of Linux (resp. Windows) adopters in the entire population;

• a estimates preference for standardization: the greater a, the more potential adopters are driven towards standardization, for instance because of compatibility issues;

• b estimates the statistical preference for global vs. local standardization.

Clearly, both a and b characterize technologies and their associated *network effects*. Typically, both a and b will be high for operating systems such as Linux and Windows. In the limit case, a technology with no associated externalities would correspond to a = 0, which would result in a uniform probability of ½ to adopt either technology, not depending any more on local and global externalities.

• α estimates the *relative* influence of previous local Windows adoptions as compared to previous local Linux adoptions;

• β estimates the *relative* influence of previous global Windows adoptions as compared to previous global Linux adoptions.

As we have outlined above, both local and global effects are different for Libre and proprietary software: they will both be stronger for Libre software. Global externalities are stronger for Linux since Linux (Libre software) improves more quickly than Windows (proprietary software), as evolutions of Linux are constantly accessible on-line and for free while Windows users have, except for

14 See also Dalle (1998ab).

important bugs, to wait for years until a new, not completely compatible version is available, which they have also to *buy*[15]. We consider here that there are no differences in access conditions for Windows and Linux, thanks to the Internet and to ancillary business which distribute Linux packages to classical retailers. And local externalities are also stronger since Linux users are more prone to proselytism than Windows users, as Linux users advertise the quality of their technology, while also stressing all the problems experienced by Windows. Therefore:

$$\alpha < 1; \beta < 1$$

A random potential adopter will adopt Linux more frequently than Windows when the number of previous adopters of Linux and Windows is the same in its neighborhood (or globally). To put it differently, the propensity to adopt Linux will be higher than the propensity to adopt Windows with a similar level of either local or global positive externalities.

3.3. Results

Each potential adopter is associated with a local neighborhood (a subset of other potential adopters): we assume here for tractability reasons that neighborhoods are organized as in a 2-dimensional torus - the « interaction structure » -, i.e. that all adopters have 4 local relevant neighbors[16]. We simulate technological trajectories by choosing a random adopter at discrete times, i.e. a position on the interaction structure and an adoption behavior given local and global environment, according to the statistical demand function outlined above: we repeat the algorithm up to 100 000 times and we measure the (possibly infinite) time needed for Linux to reach a 70% market share. Initial conditions are a uniform adoption of Windows[17]. We repeat the entire process for each α and β between 0 and 1 with step 0.1. Figure 1 below gives simulation results for a = 2 and b = 1, which seem appropriate values for a technology highly sensitive to standardization phenomena such as operating systems.

---

15 Microsoft, being a monopolist, has specially low incentives to re-invest increasing returns of adoption: an explanation perhaps for a somewhat slower improvement rate and for more opportunistic release strategies than for other software producers.

16 Here for simulation simplicity a 30x30 2-dimensional torus (900 potential adopters).

17 A further improvement if compared with Arthur's (1989) model which limits itself to considering competition between two new born technologies.

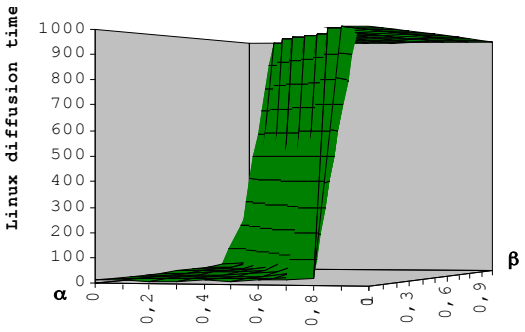

Figure 1: Linux diffusion time (static α and β parameters)

Results exhibit a phase transition, i.e. α and β behave as state parameters associated with a sharp discontinuity in diffusion regimes. Above critical values of α and β diffusion is infinitely long – it will almost never occur in economic time – whereas below them diffusion is almost sure and even fairly rapid. As a consequence, Linux wins when it benefits from *significantly stronger* global and local externalities than for Windows[18].

Similar results obtain when Microsoft answers the Linux threat by lowering its prices or for instance by shifting its strategy towards renting instead of selling software[19]. To simulate this, we consider that β is significantly increased during the diffusion process as soon as the number of Linux users reaches a given threshold defined as a percentage of the entire potential adopter populations[20]. Figure 2 presents results with such a protocol: as the percentage of Linux adopters reaches 5%, β is increased by 0.2, therefore a strong reaction at a very low threshold (a = 2 and b = 1 here again, plotted variables α and β correspond

---

18 It is interesting here to note that a minor technological innovation might sometimes prove powerful enough to replace a dominant standard as long as it is associated with a superior economic model. Taken as a technology, Linux is indeed not really superior to Windows. Here superiority does not come from pure technological arguments, but from a superior *economic model* i.e. from non-technological aspects. In any case, technological trajectories are thus characterized by persistence (Foray, 1997) phenomena rather than by perfectly irreversible path-dependence (David, 1985). A way to analyze this is to suggest that diffusion creates *endogenous thresholds*: new technological candidates have to be « sufficiently better », be it technologically or economically, to get rid of existing standards, i.e. above a given threshold and to become *major* innovations, whereas *minor* innovations fail to pass this threshold. As a consequence, minor innovations will most often to be rendered compatible with existing standards as they would otherwise have almost no chance of diffusion: the existence of endogenous diffusion thresholds creates incentives for minor innovations not to challenge an existing standard but to improve it (Dalle, 1998b).

19 Microsoft strategy is indeed from time to time said to evolve towards software rental with continuous bug correction i.e. with no more “versioning”: the so-called “.NET” strategy might be a step in that direction.

20 Another simulation methodology would have been to consider a continuous evolution of β. We prefer this one as major shifts will most probably be rare, if only because Microsoft also has organizational routines which render such changes very difficult and painful to manage.

to their initial values). Still, Linux diffuses in a very similar way to what happens without any such reaction or evolution. Compared with previous results, we find that diffusion is roughly similar to what would happen with an initial value of β such as $\beta = \beta+\delta$ with $\delta < 0.1$.

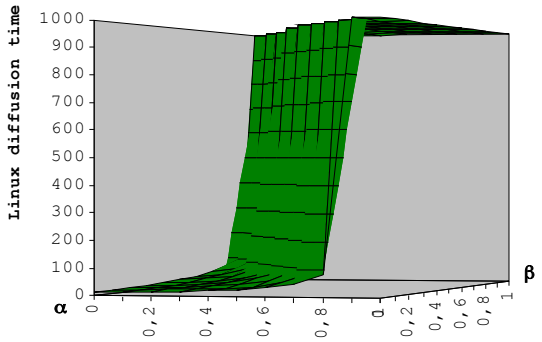

Figure 2: Linux diffusion time when β varies during diffusion

Yet similar results obtain when α is increased during the diffusion process i.e. would Linux users partly cease to be zealots after a while. In both cases, we indeed find a strong path-dependence effect (David, 1985): results appear as weakly sensitive to possible variations of state parameters during diffusion processes. Once diffusion has started, it is difficult to stop it. To put it differently, (small) historical events do matter a lot, and proselytism induced by some sort of Microsoft-phobia may have played a similar role for Linux as early technological constraints once played for the QWERTY keyboard. Proselytism might disappear soon, but it might also have permanently oriented the diffusion path, whatever Microsoft reactions might be. Idiosyncratic historical conditions might have given sufficient initial momentum for Linux to diffuse.

3.4. Consequences

A major consequence of these results if that two major conditions have to be verified, at least at the beginning of the diffusion process, for Linux to defeat Windows. First, communities and associated ancillary businesses have to be effective enough to allow for a sufficiently quick improvement rate; and second, local network effects have also to support Linux strongly enough.

The first condition (high β) itself relies on two complementary aspects: the capacity of developer communities to attract a sufficient number of good kernel developers and numerous obscure developers, and their ability to organize efficient development processes, an issue which can be significantly addressed by the existence ancillary business firms. Indeed, these companies not only help create incentives for kernel developers and provide a considerable amount of man-year worth of obscure development, but they also contribute to organize efficient development processes, for instance by helping developer communities to select the best issues to solve first and generally by providing extra organization to Libre software communities. The victory of Linux, and generally

of Libre software, therefore depends on its ability to merge independent developer communities and business-oriented companies in an efficient organizational way.

As for the structure of these communities, the existence of a limited core of kernel developers, generally associated with cooptation rules, indeed appears as a *necessary* condition for Libre software to develop. More than a proportion of kernel and obscure developers, the relevant question here is mainly about the existence of such a core of kernel developers. Then the existence of numerous obscure developers is a *sufficient* condition, not only for work to be done, but also to allow for further recruitments. There would be no Linux without Linus Torvalds, Alan Cox, and a limited subset of other kernel developers, but there would be no Linux either if they would be alone[21].

But the eventual victory of Linux depends also on the proselytism of its (early) adopters (high α): a condition easily verified for Linux, more easily perhaps than the previous one, but a condition which might conversely be quite difficult to account for other pieces of Libre software. Although more studies are needed here before we could generalize these last results to other examples of Libre software[22], Linux might still be more sustainable than many other pieces of Libre software because they might lack sufficient support from their early adopters.

Since both conditions are specially critical at the beginning of diffusion processes, we finally conclude that the development and the sustainability of Libre software, measured by its ability to become competitive against proprietary products, critically depend on an *initial momentum* issue which deals both with the organization of development processes within communities and business firms, and with support by early adopters.

4. Conclusion

If we were some day to consider that Libre alternatives to proprietary products could needed to improve social welfare, for instance in the case of operating systems, middleware, and generally for software which is widely used and strongly subject to strong compatibility issues and network effects, then our conclusions imply that there might be some need for a public intervention to help foster the initial momentum of Libre solutions.

A major question for public agencies, and an agenda for future research, would then be to determine more precisely which software would be suitable for such issues, and notably to avoid opportunistic strategy by some software producers which would certainly try to adopt Libre models just to benefit from public subsidies and therefore to lower their R&D costs. In this respect, the existence of a dominant proprietary standard would certainly be a first-order condition.

---

21 See e.g. http://www.wired.com/wired/archive/9.10/.

22 It would is necessary to analyze how these results vary for types of Libre software other than operating systems, for instance considering different values of a and b.

A second, related question would be to determine what kind of actions could be undertaken to help launch Libre software development projects, i.e. both to create new or to develop existing Libre software communities, and to replicate local network effects in Libre software diffusion without relying on cultural feelings. Once these issues have been solved, Libre software might become a excellent cure for network externalities.

Then, Libre software, born as a fad, would actually turn into an economic institution. In this respect, let us simply recall that both the scientific institution with the associated convention of « Open Science », and the intellectual property institution have also been born in somewhat similar and yet surprising circumstances: as a matter of fact, « lettres de patentes » were once used in Renaissance France to attract foreign inventors so that they would import technologies which already existed elsewhere (David, 1993), whereas scientists were originally "devoted" to raising princes' reputation, which implied that they render their discoveries public (David, 1995)… In a way, we might perhaps better start considering Libre software positively, against today's still prevailing skepticism, since it might be giving birth not only to an original incentive structure within Libre software communities but also to a new and general economic model for the software industry, a model whose consequences have yet largely to be understood.

Bibliography

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

Cohen M. D. (1983), Conflict and Complexity: Goal Diversity and Organization Search Effectiveness, American Political Science Review, 78 : 435-451.

Cohendet P., Créplet F., Dupouët O. (2000), Communities of practice and epistemic communities : a renewed approach of organizational learning within the firm, paper presented to WEHIA 2000 Conference, Marseille, June.

Cowan R., Foray D. (1997), The economics of codification and the diffusion of knowledge, Industrial and Corporate Change 6: 595-622.

Dalle J.-M. (1995), Dynamiques d'adoption, coordination et diversité, Revue Economique, 46: 1081-1098.

Dalle J.-M. (1997), Heterogeneity vs. externalities: a tale of possible technological landscapes, Journal of Evolutionary Economics 7: 395-413.

Dalle J.-M. (1998a), Heterogeneity and rationality in stochastic interaction models, in Cohendet P., Stahn H. (eds), The economics of networks: behaviors and interactions, Springer Verlag: Berlin, pp. 123-145.

Dalle J.-M. (1998b), Local interaction structures, heterogeneity, and the diffusion of technological innovations, in Orléan A. & Lesourne J. (ed), Self-organization and evolutionary approaches: new developments, Economica: Paris, pp. 240-261.

Dalle J.-M. (2000), Patents and anti-patents: institutional means to enhance creativity, paper presented to IFREDE-E3i Conference held in Bordeaux, October.

Dalle J.-M., Jullien N. (2000), NT vs. Linux, or some explorations into the economics of Free Software, in Ballot G., Weisbuch G. (eds), Application of simulation to social sciences, Hermès, Paris, pp. 399-416.

Dasgupta P., David P.A. (1994), Towards a new economics of science, Research Policy 23: 487-521.

David P.A. (1985), Clio and the economics of QWERTY, American Economic Review (Papers and Proceedings) 75: 332-337.

David P.A. (1988), Putting the past into the future of economics, Technical Report 533, Institute for Mathematical Studies in the Social Sciences: Stanford University.

David P.A. (1993), Intellectual property institutions and the Panda's thumb, CEPR publication n°287, Stanford University.

David P.A. (1995), Reputation and agency in the historical emergence of the institutions of 'Open Science', paper presented to the National Academy of Sciences Colloquium on the Economics of Science and Technology, held at the Beckman Center, UC Irvine, 20-21 October 1995.

David P.A., Foray D. (1995), Accessing and expanding the science and technology knowledge base, STI Review 16 : 13-68.

David P.A., Foray D., Dalle J.-M. (1998), Marshallian externalities and the emergence and spatial stability of technological enclaves, Economics of Innovation and New Technology 6: 147-182.

Durlauf S.N. (1993), Non-ergodic economic growth, Review of Economic Studies 60 : 349-366.

Foray D. (1997), The dynamic implications of increasing returns: technological change and path-dependent inefficiency, International Journal of Industrial Organization 15: 733-752.

Harhoff D., Henkel J., Von Hippel E. (2000), profiting for voluntary information spillovers: how users benefit by freely revealing their innovations, mimeo, 26p.

Jullien N. (1999), Linux, la convergence du monde Unix et du monde PC?, Terminal 80-81: 41-70.

Kirman A.P. (1993), Ants, rationality and recruitment, Quarterly Journal of Economics 111: 137-156.

Lakhani K., Von Hippel E. (2000), How Open Source software works : 'free' user-to-user assistance, MIT Sloan School of Management WP #4117.

Lerner J., Tirole J. (2000), The simple economics of Open Source, mimeo, 38p.

Romer P. (1993), The economics of new ideas and new goods, Proceedings of the World Bank Annual Conference on Development Economics 1992, World Bank, Washington DC.

Von Hippel E. (1988), The sources of innovations, MIT Press.