Competition and Adoption of Open Source and Proprietary Software

Eric DARMON ♣ Thomas LE TEXIER ♣

Dominique TORRE ♣

♣ *GREDEG - CNRS - University of Nice Sophia-Antipolis*
250, bd. Albert Einstein, F-06560 Valbonne Cedex
E-mail: {darmon ; letexier ; torre}@idefi.cnrs.fr

First Draft – Please do not quote without authors' permission
June 2005

Abstract

In this paper, we study the conditions ruling the diffusion of an open source software as opposed to a proprietary one. We first define the characteristics attached to these two types of software: going beyond a distinction solely based on license costs, the two software also differ according to their usage costs (adoption costs, existence of customized functionalities). Besides, we distinguish two categories of users, namely "end-users" and "users-developers", according to their ability to contribute to the development of the OS software. In this framework, we characterize the Nash equilibria defining the adoption of open source and proprietary software. In particular, we show that the adoption of the OS software may exhibit path dependency (i.e. sensitivity to the size of the population of early adopters). Starting from identical initial conditions (adoption costs, magnitude of the externalities generated by the community of developers and users, etc.), a winner-take-all competition arises between the two software which may lead to the crowding out of one of the two software. Such multiple equilibria can be only imperfectly controlled by the commercial firm. The OS project indirectly acts as a "incumbent": to avoid being locked in a non-favourable outcome, the commercial firm may be incited to invest more on quality and/or to charge smaller licence costs.

Key words: Open Source – Commercial Software – Increasing returns – Community of developers
***JEL* Codes :** D3-D4-L1

1 Introduction

Unlike their proprietary counterparts, open-source (OS) software is based on particular license terms which allow any user to freely "*use, modify and redistribute*"[1] the original software[2]. From an economic point of view, one key issue is to find out whether this original production and distribution mode is viable in the long run and could establish itself as a credible alternative to proprietary software. Empirical studies reveal that the success of OS softwares is very diverse: in the last few years, the OS model has rapidly spread in technical areas (cf. Apache, Sendmail, PHP) where most users are expert (*e.g.* developers, system administrators, webmasters). In other end-users oriented areas, the OS model shows ambiguous performances. For example, Linux starts to establish as a competitive alternative to proprietary operating systems, while LATEX (an open-source word processor) has not managed to expand to a large community of users despite it has been introduced before other proprietary solutions (cf. Gaudeul (2003a)). In all these examples, we can identify several disparate causal factors. Some of them are structural (type of users, existence and quality of a proprietary counterpart). Others are more historical (unsuited leadership of the open source software, lock-in effects). All these factors partially explain why open-source software may succeed in some cases and fail in others. One could integrate these factors into two inter-related topics: *i*) *production* of the OS software i.e. developers' individual incentives to participate in OS communities and *ii*) *diffusion* of the OS software i.e. users' individual incentive to adopt open source software.

On the developers' side, contributors' motivational aspects have been widely analyzed by both economists and sociologists. Indeed, according to Lerner and Tirole (2001), the apparent "non-profit" and altruistic character of the contributions has to be downplayed and partially replaced by more selfish objectives (carrier concerns, prestige as evidenced by the hierarchical structure of communities). Following these authors, Mustonen and Leppämäki (2003) present a situation in which programmers' involvement in open source projects is considered as a signal for their potential employers and the development of the open-source project can either harm the proprietary software (substitute products) or benefit from it (complementary products). This signal hypothesis has been empirically tested and corroborated by Apache developers (cf. Hann *et al.* (2002)). Furthermore, Dalle and Jullien (2003) emphasize the crucial role of institutions (as e.g. the *copyleft* licence) to provide a credible commitment able to preserve such incentives. Indeed, developers' incentives are highly dependent on the type of license under which the OS is released (cf. Gaudeul (2003b) for a theoretical analysis of developers' and project leaders' incentives under the GPL, BSD and proprietary licenses). Incentives are also sensitive to the way the OS project is managed. In particular, the type of leadership (cf. Gaudeul (2003a)) and the organization of the software (modular versus centralized, see e.g. Hertel et al. (2003)) should be stressed.

On the users' side, the adoption issue has attracted less attention. Dalle and Jullien (2003) develop a simulation model: assuming that all potential adopters initially use the proprietary software and that network externalities are both local and global, they show that the diffusion of the 'libre' software goes through a phase of transition associated with a lock-in effect. As a consequence, they point out the crucial role of early adopters on the success or the failure of the open source project. However, their analysis is based on the premise that, everything being equal (i.e. when the two software have the same share of the global and local markets), the quality of the OS software is better than that of the proprietary one. In a close framework, Bonaccorsi and Rossi (2003) add the possibility for users either to adopt the two kinds of software simultaneously or separately. While the network externality is strictly global, they differentiate users according to the benefit they derive from the adoption of the open-source software. The intrinsic utility attached to the open-source software is probabilistic and several distributions are tested (normal, uniform, exponential biased beliefs against the open source software, exponential biased beliefs against the proprietary software). With 'moderate' network effects, they conclude that both type of software coexist with all forms of distribution (except for the exponential distribution biased against the open-source software, which blocks the diffusion of Open Source). However, the coexistence of the two technologies may be partly caused by the fact that agents can maintain the two types of software simultaneously without additional costs. This feature does not hold in the long run for software which are not

[1] See Cremer and Gaudeul ((2004)) and von Krogh and von Hippel (2003) for a general presentation of Open Source.

[2] GPL (*General Public License*) and BSD (*Boston Software Development*) are the most famous open-source licenses but it can be shown that there exists a continuum of license ranging from pure open-source software to proprietary one (see West (2003), Muselli (2004) and Zimmermann and Jullien (2002) for an analysis of such mixed developement strategies and Lerner and Tirole (2002) for an empirical survey). GPL and BSD licenses differ on the following point: any software which draws on GPL software, has to preserve the original licence terms, while BSD license terms allow a developer to freely change the type of the licence, e.g. transforming it into a proprietary or a GPL license.

perfectly compatible[3]. In a static context, Mustonen (2003) studies the interaction between open source and proprietary software through the allocation of developers' time: developers can choose to allocate their time either by being hired by the proprietary software company (direct revenue) or by participating in an OS project and thus acquiring experience and signalling their quality to future employers (indirect revenue). Depending on the magnitude of the open source adoption cost incurred by users, the commercial firm can crowd out the diffusion of the open-source software by proposing higher wages (which discourages developers to participate in the open-source community) or by increasing the quality of its software (vertical differentiation). However, this model assumes that developers are not users and consequently that developers' participation is only motivated by signalling motives. As it should be stressed, another important motive to participate in OS community is to develop a software customized to the developers' own needs (innovation-led users).

Tackling the adoption issue, it seems important to stress the heterogeneity of users' needs. Users are first heterogeneous regarding their preferences i.e. two users may not use a software to perform the same task. In particular, we develop a model which is able to take into account two motives for OS adoption: i) difference in license cost when a user can substitute a proprietary software to an OS software and ii) adoption of OS as the proprietary software does not provide specialized functionalities. Along with this heterogeneity in preferences, users are also different according to their contribution to the open source software. We should at least consider two distinct populations: *developers-users* (cf. Franke and von Hippel (2003)) and *end users*. End users can orient the future development (*e.g.* by asking for new functionalities) but are not able to contribute directly to the development effort. Developers-users may use the software for their own needs (or for equivalently for the needs of their company) and are able to write new lines of code and customize the software as they need new functionalities. The relative size of these two populations obviously depends on the type of software: highly technical software may primarily concern users-developers while operating systems or office suites may concern a larger set of final-users. In turn, this distinction affects the economic incentives that drive each population to adopt the OS software or not. In other terms, the presence or the absence of one type of users alters both the benefits (development of functionalities) and costs (presence of a wide community of users to help new users and lower adoption and implementation costs). In this paper, we consider the adoption issue by explicitly taking into account the coexistence of heterogeneous users (i.e. users with different programming ability and differentiated preferences). As a twin issue, we propose to evaluate the welfare generated by the introduction of OS software. This last question seems particularly relevant considering the recent initiatives to promote open source adoption[4]. Indeed, it is straightforward to mention that individual and collective welfare may diverge in the presence of network externalities which are typical of software goods (cf. Shapiro and Varian (1999))[5]. To be fully appropriate, such policies then need a comprehensive framework able to account for such externalities at a more macroscopic level. Schmidt and Schnitzer (2003) survey some of these policy-related topics. With a simple model (horizontal product differentiation and three populations of users, a 'mobile' population and two captive ones), they show that increasing the share of OS captive users can harm the welfare of other populations. This model shares several features with our model, but we do not suppose here that some users are exogenously captive to one kind of software and add the important distinction between end users and developer-users.

This paper is organized as follows: Section *2* presents the model. Section *3* reviews the Nash equilibria. Welfare properties are analyzed and discussed in Section *4*. Section *5* concludes.

2 The Model

2.1 End-Users and Developers

There are two categories of users: *developers* (indexed by D, in proportion μ of the total population) and *end-users* (indexed by *EU,* in proportion $1-\mu$ of the whole population). The mass of the whole population is normalized to 1. End-users and developers differ according to two elements: first, developers are able to develop

[3] For example, while switching from the proprietary suite *Microsoft Office* to the GPL *Open Office*, a user may loose functionalities (e.g. macro-commands). It may then be more costly to maintain two versions of the same file with the same quality than to adopt only one of the two software (either open-source or proprietary).

[4] See, e.g. China's plan to equip its central administration or Taiwan's initiative to promote the development of Open Source programs.

[5] See Schmidt and Schnitzer (2003) for a survey of all policy-related topics. With a simple model (horizontal product differentiation and three populations of users, a 'mobile' population and two captive ones), they show that increasing the share of OS captive users can harm the welfare of other populations.

new functionalities and customize an OS software by adding new lines of codes, while end users are not. In other words, as the source code is open, a developer is able to develop new functionalities for his own benefit. Once these functionalities are developed, there is no cost in releasing them to the whole community[6]. Besides, an end-user will freely benefit from the experiences and releases of developers. Second, End users and developers do not face the same adoption costs. Developers are generally expert users (engineers, computer scientists) and are used to adopt new software. On the contrary, end users face high initial costs while adopting a new software. It seems reasonable to assume that these costs are generally higher when adopting an OS than those incurred when adopting a commercial one. One reason for this is that OS software are primarily developed and leaded by expert users who cannot devote much time to make tutorials, FAQ, numerous user-friendly interfaces, etc[7].

For that reasons, we will suppose that developers do not face any adoption cost while end users do. Even inside the population of end users, some agents are more expert than others. That is why we will suppose that the adoption cost is heterogeneous among this category of agents End-users are heterogeneous in this respect and face an adoption cost uniformly distributed between 0 and $\overline{c}^{EU}$.

2.2 Commercial and Open-source software

The two software need to fill two types of functionalities i.e. generic and specialized functionalities. One may think of e.g. a typesetter. Generic functionalities are to print, edit (cut and paste, change font), exchange a document. Specialized functionalities are for example to insert external objects (equations, media files), or to make dynamic updating inside the document (bibliographic items, link with databases, etc.). Users are heterogeneous regarding the second component. We will thus suppose that they are uniformly distributed on a unit circle: each agent is located on a point and needs the functionality related to that point. Users may choose *i)* to adopt a proprietary (closed source) software; *ii)* a GPL-open source software; or *iii)* to adopt neither of the two softwares. In the last case, one can suppose that users can manage to do the same task either "manually" or with a combination of previously acquired software (i.e. without costs). This default option leads to a null payoff.

- *The strategy of the commercial firm*

We suppose that a single commercial firm produces the proprietary software. Because the software is essentially a digital good, its reproduction costs are negligible and will thus further assume them to be null (and independent with the number of adopters). As mentioned before, the proprietary software needs to satisfy both generic and specialized functionalities. For its software to be eligible, we will suppose the firm needs to develop *all* generic functionalities. Let us go back to the typesetter example: no user will be willing to pay for a typesetter unable to print. For that reason, this development cost can be analysed as a fixed cost. Conversely, the extent of the specific needs covered by the proprietary software is controlled by the firm. By selecting a more or less important research effort on R&D, it can choose to develop more or less specialized functionalities and to cover a more important fraction if the unit circle (noted q). As a consequence, the production costs of the proprietary software are fixed relatively to the number of users (digital good, no reproduction cost), but variable relatively to the extent of the provided functionalities. This is one of the control variables for the firm. The firm needs to invest in R&D to develop these functionalities. We will thus assume that there is a cost $g(q)$ (increasing with q) to provide them.

Let us denote by q , $(q \in [0,1])$, the extent of functionalities offered by the firm, defined as a proportion of the unit circle covered by the selected software quality, and by c_L ($c_L > 0$), the license fee paid to the firm by each user. The commercial firm chooses a couple (q^*, c_L^*) maximizing its profit given by Equation (1) :

[6] We restrict our analysis to GPL (General Public License) licenses. Other types of licenses (as e.g. BSD) allow any user or developer to commercialize its own software derived from the initial software. Consequently, this other type of license raises an additional incentive problem for developers which may be inclined to "close" the code of their customized software instead of making it freely available to other users. This specific case could be the subject of an extension of the present work.

[7] It could be argued that users may incur an adoption cost even when adopting a commercial software. However, because user-friendliness is an important factor for adoption and purchase of a software, commercial firms devote important marketing efforts to maximize the usability of their software. To clearly delineate OS from commercial software, we can thus neglect adoption costs for the proprietary software.

$$\pi = n^P c_L - g(q) \quad (1)$$

where n^P $(n^P \in [0,1])$ denotes the proportion of users of the proprietary software and $g(q)$ is an increasing function such that $g(0) = 0$.

- *The utility generated by the proprietary software*

The ability of the commercial software (and of the OS also) to fill the generic functionalities strongly depends on the total number of adopters (see e.g. Shapiro and Varian (1999)). Let us consider again the example of the typesetter: most users need to exchange files with other users, read and edit files created by others, etc. More generically, this network externality is motivated by several factors, such as compatibility between users, availability of complementary services and software, *etc*. To account for it, we will suppose that the utility associated to such functionalities is closely dependent on the number of its users. Consequently, we assume that the utility generated by the general needs $f_1(n^p)$ positively depends on the adoption level expressed by n^P (with $f_1'(.) > 0$, $f_1(0) = 0$ and $f_1(1) = k_1, (0 < k_1 < \infty)$.

As the source code of the proprietary software is closed, no user can develop his own specialized functionalities. The utility generated by the commercial software then depends on whether the commercial firm has developed the desired specific functionality. Users whose functionalities are covered by the commercial software get an additional utility equal to d $(\text{with } 0 < d)$, 0 otherwise. When adopting a software, a user cannot fully anticipate which specialized functionality he will need in the future (i.e. his location on the circle). Consequently, he can only consider the extent of specificity covered by the commercial software and try to anticipate the future utility of the commercial software. We will also suppose that users are risk-neutral: as a consequence, the expected utility of potential users relatively to the specific uses of proprietary software is the weighted average of their respective *ex post* utilities according their location, inside or outside the set of the functionalities effectively covered by the commercial software. Let d corresponds to the utility generated by the specific uses of proprietary software. Once considered that the firm develop a proportion q of all the specialized functionalities, the *ex ante* utility derived from the specialized functionalities is thus equal to $q(d) + (1-q)(0) = dq$.

Once deduced the licence cost c_L charged by the firm, the net utility derived from the adoption of the proprietary software can be expressed as follows (equation 2):

$$u_i^p = f_1(n^p) + dq - c_L \quad (2)$$

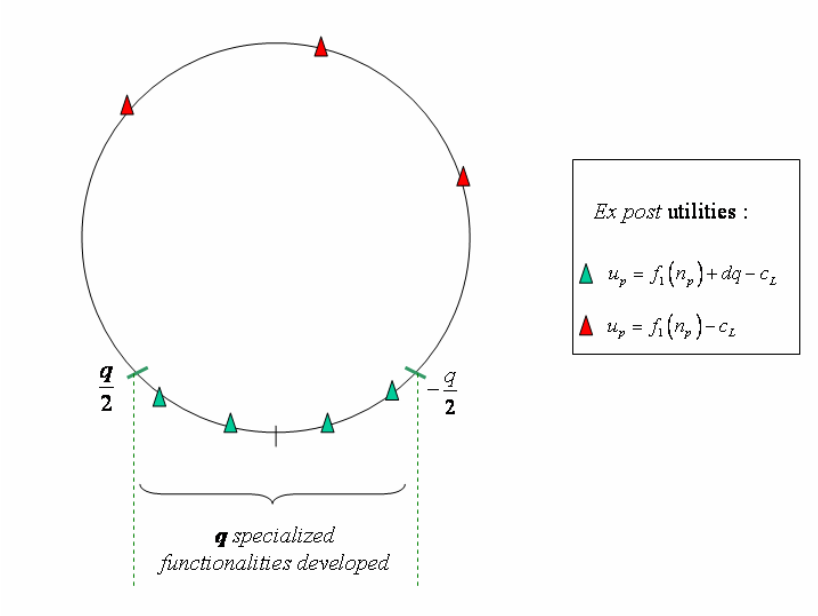


Fig. 1: The utility of the Commercial Software

- *The utility generated by the OS software*

When adopting the OS software, there is no license fee. Similar to the commercial software, we will suppose that the software need to fill all the generic functionalities to be eligible[8]. As previously, these generic functionalities are subject to a compatibility externality (noted again f_1). Besides, developers may, in the case of OS software, freely modify and customize the initial software. The modular organization of most recent open source software favours such customization so that open-source developers may add functionalities that will then benefit all users. The OS software is maintained and managed by developers. Hence, the utility derived from specialized functionalities is highly dependent on the number of contributors (noted n_D^{OS})[9]. In other terms, the more developers contribute to the code, the more functionalities are added, and the higher the utility of the open source-software is. This positive link will be described by a positive externality (noted f_2) that benefits the whole population (users and developers). We will capture such influence of the size of developers' population on the OS individual utility by a positive externality noted $f_2\left(n_D^{OS}\right)$ with n_D^{OS} denoting the size of the community of developers ($f_2'(.) > 0$, $f_1(0) = 0$ and $f_2(0) = 0, f_2(1) = k_2, (k_2 > 0)$.

The adoption cost c_i^{EU} $\left(0 \le c_i^{EU} \le \overline{c}^{EU}\right)$ of any user $i\left(i \in [0,1]\right)$ is depicted by Equation (3). Developers and end-users are ordered according to the size of their adoption costs. Consequently, as $i \in [0, \mu]$, we will conventionally posit that user i is a developer and his adoption cost is null. As $i \in [\mu, 1]$, user i is an end user and is defined in such a way that $c_i^{EU} = \left(\frac{i-\mu}{1-\mu}\right)\overline{c}^{EU}$ (cf. figure 2).

[8] We may suppose that these functionalities are "ex ante" developed by "kernel" developers or by the project manager, so that the OS exists, once all these functionalities are fully available (release of the "1.0 version").

[9] We will assume that once a developer adopts the open source software, he actively contributes to the code and does not act as an end-user. This simplification implies that n_D^{OS} is equal to the size of the population of developers who adopt the OS.

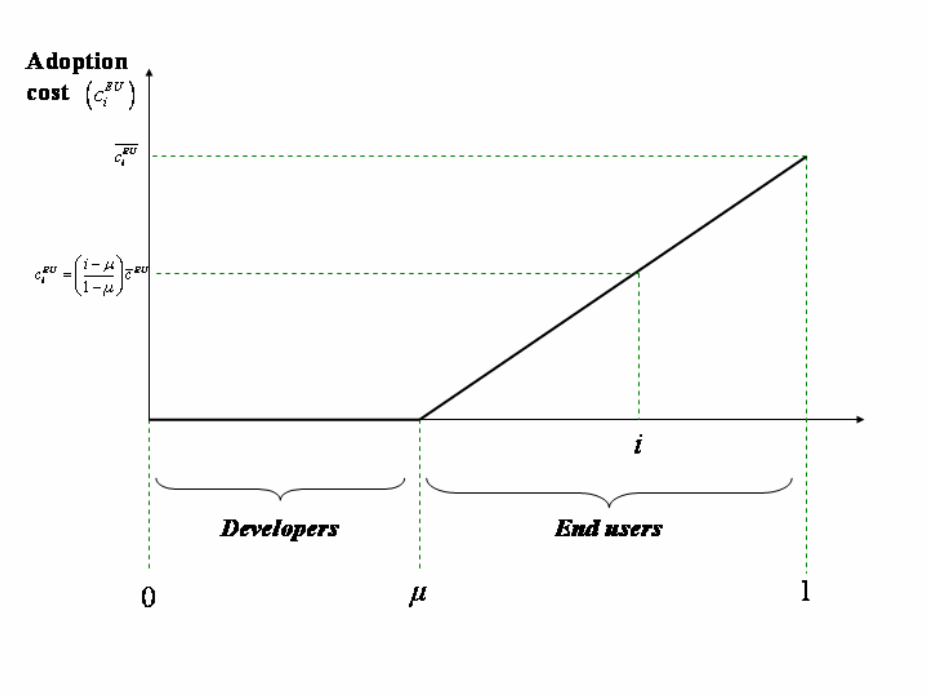


Fig. 2: Adoption costs of the OS Software

The utility generated by the open source software for an agent i ($i \in [0,1]$) can now be expressed as follows:

$$u_i^{OS} = f_1\left(n^{OS}\right) + f_2\left(n_D^{OS}\right) - c_i^A \text{ with } c_i^A = \begin{cases} 0 \text{ when i} \in [0,\mu] \\ c_i^{EU} \text{ when i} \in [\mu,1] \end{cases} \quad (3)$$

2.3 The Sequential Game

The firm has complete but imperfect information about the composition and the possible strategies of the potential adopters of its software: it knows the different characteristics of its potential users $(\mu, \alpha_1, \alpha_3, \ldots)$ but can imperfectly anticipate their behaviour. It is reasonable to suppose that the price and the quality of a software are not decisions that can be renewed frequently. Under this assumption, we suppose that firms' and users' choices are sequential: the firm makes its choices initially and users reply subsequently to firms' strategy by formulating their own adoption choice. The structure of the sequential game is depicted by Figure 2:

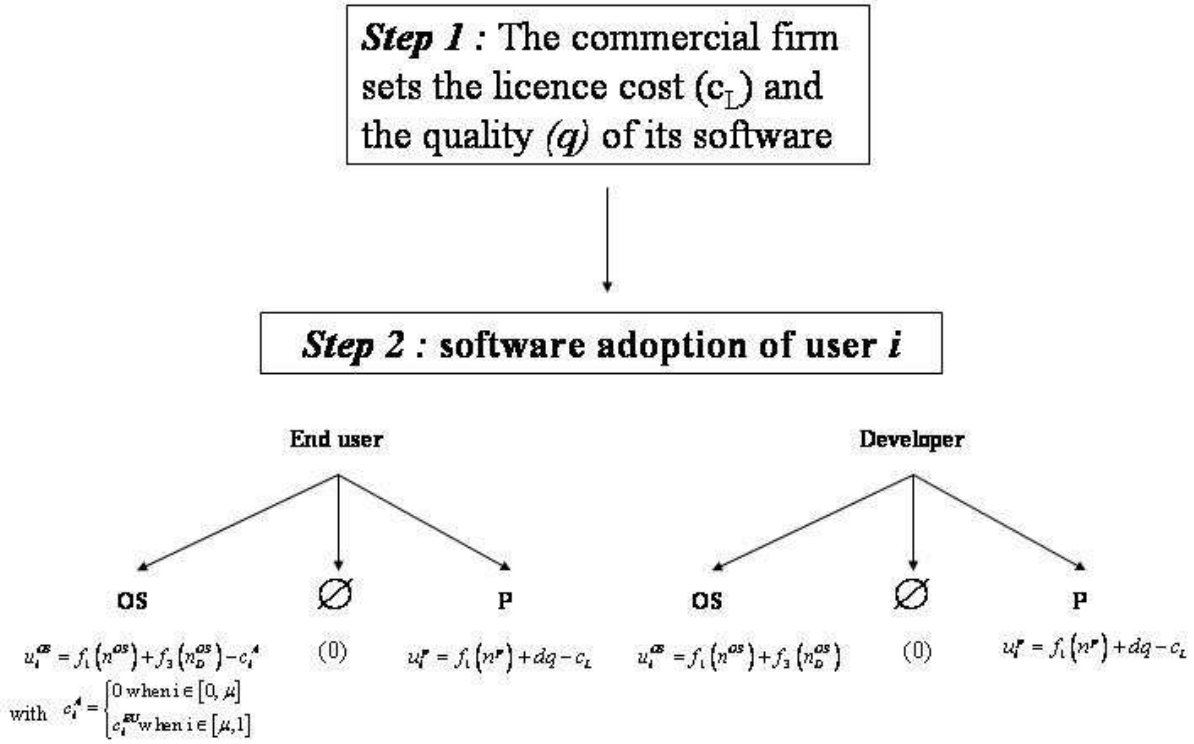


Fig. 3: Structure of the sequential game (OS = Open Source Software adoption ; P = Proprietary Software adoption ; ∅ = No adoption)

At *Step 1*, the firm chooses the extent of functionalities q^* and the level of its license fees c_L^* that together maximize its expected profit. At *Step 2*, each user observes these two variables, compares its respective utilities and chooses between the proprietary software (Strategy "*P*"), the OS software (Strategy "*OS*") or no adoption (Strategy "∅"). In doing so, agents expect the level of the relevant populations of commercial software customers, end users and developers of the Open Source software. In the (frequent) cases where these expectations would have been erroneous, they choose as subsequent prediction the currently observed size of these three populations (purely static expectations).

3 Nash Equilibria of the Sequential Game

We solve the game by backward induction : we consider first the second sub-game (adoption decisions of developers and end-users for a given strategy of the firm). Solving backward, we then determine the profit maximizing strategy of the firm

3.1 The Second Step

3.1.1 General case

Let us thus consider a couple (q, c_L). Once we note the expectations $\tilde{n}^P$, $\tilde{n}^{OS}$, $\tilde{n}_D^{OS}$ relative to the respective sizes of the commercial software customers population, the total population of OS end users and the population of OS developers, the choice of any user $(i,\ i \in [0,1])$ is as follows:

- Choosing *P* if $f_1(\tilde{n}^P) + dq - c_L \geq \sup\{0, f_1(\tilde{n}^{OS}) + f_3(\tilde{n}_D^{OS}) - c_i^A\}$
- Choosing *OS* if $f_1(\tilde{n}^{OS}) + f_2(\tilde{n}_D^{OS}) - c_i^A \geq \sup\{0, f_1(\tilde{n}^P) + dq - c_L\}$
- Choosing $\varnothing$ if $0 \geq \sup\{f_1(\tilde{n}^P) + dq - c_L, f_1(\tilde{n}^{OS}) + f_2(\tilde{n}_D^{OS}) - c_i^A\}$

Each agent replies to each expected level of $\tilde{n}^P$, $\tilde{n}^{OS}$ and $\tilde{n}_D^{OS}$, by formulating its optimal choices. Let us assume conventionally that if they expect the level $\tilde{n}^{OS}$ for the OS users, they will consistently suppose that this population is only formed by developers if $\tilde{n}^{OS} \leq \mu$ while it also integrates in supplement the end-users whose adoption costs are the lowest if $\tilde{n}^{OS} > \mu$. From the aggregation of these optimal decisions, the effective proportions of proprietary software users, of developers and end-users of Open Source software n^P, n^{OS} and n_D^{OS} can be deduced. Hence, for each couple (q, c_L), the condition $(n^P, n^{OS}, n_D^{OS}) = (\tilde{n}^P, \tilde{n}^{OS}, \tilde{n}_D^{OS})$ defines the equilibria of the sub-game corresponding to *Step 2* of the game.

Without any specification of the functions f_1, f_2 and g, the following propositions can be established in the general case:

Proposition 1: Whatever the pair (q, c_L) selected by the firm at the Step 1 of the game, it exists at least one Nash equilibrium associated to Step 2 subgame. *(Proof, see Annex 1)*

Without any specifications on the functional forms of $f_1(.), f_2(.)$ and of the development costs $g(.)$, we have no additional precision about the nature of the equilibrium position(s) of the second step sub-game. However, such precisions can be obtained with a slightly weak assumption on the economic "viability" of the firm. For the firm to be economically viable and the proprietary software to survive, one minimal condition is that it there exists at least one pair (q^+, c_L^+) providing a positive utility to the users of commercial software. Under this minimal condition, we can prove the following result:

Proposition 2: Suppose that the pair (q, c_L) selected by the firm at the Step 1 is such that it provides a positive utility to the users of commercial software at least for one distribution of agents among P, OS and $\varnothing$. Then the triplet $\{n^P* = 1, n_D^{OS}* = 0, n^{OS}* = 0\}$ corresponding to the full adoption of the commercial software is a Nash equilibrium of Step 2 sub-game. *(Proof, see Annex 1)*

Proposition 2 establishes that under very weak assumption, there *always* exists at least one equilibrium such that the Open Source software is crowded out. At this equilibrium, all users adopt the commercial software. The needed assumption is quite weak since when this assumption is not verified, it would mean that the firm is unable to impose its software even to one fraction of the whole population of users.

Other equilibria may exist. An elementary example of such multiplicity could be as follows: let us consider the (limit) case where $c_i^A = 0, \forall i$. In this case there exists a range of variation of q^+ and c_L^+ (q^+ sufficiently high and for c_L^+ sufficiently low) such that $f_1(1) + dq^+ - c_L^+ \geq 0 \geq f_1(0) + dq^+ - c_L^+$. We then verify that $f_1(1) + f_2(\mu) - c_i^A > \sup\{0, f_1(0) + dq - c_L\}$ whatever i. The outcome $\{n^P** = 0, n_D^{OS}** = \mu, n^{OS}** = 1\}$ is then a second equilibrium of the second sub-game. By continuity and by the fact that *OS* is for all users strictly preferred to *P* when $\{\tilde{n}^P = 0, \tilde{n}_D^{OS} = \mu, \tilde{n}^{OS} = 1\}$, the same result is maintained when $c_i^A = \varepsilon_i\ (\forall i)$ with ε_i as close as possible to 0 (but $\neq 0$). Then, the equilibrium $\{n^P** = 0, n_D^{OS}** = \mu, n^{OS}** = 1\}$ exists for a not-empty range of variation of $c_i^A\ (\forall i)$. When $c_i^A\ (\forall i)$ is far from 0, the result is not yet maintained. The nature of the equilibria

other than $\{n^P* = 1, n_D^{OS}* = 0, n^{OS}* = 0\}$ yet depends (when they exist) on the forms of $f_1(.), f_2(.)$ and $g(.)$. We thus need to specify these functions to obtain further results.

3.1.1 The Linear-quadratic specification

Let us consider that the functions describing the compatibility and development externalities ($f_1(.), f_2(.)$ respectively) are linear. Let us also assume that $g(.)$ is quadratic. This last (conservative) assumption implies that it is more and more costly for the firm to develop the functionalities for a large set of users. Thus, the profit of the firm falls and the utilities of the commercial and OS software can be re-expressed as follows:

$$\pi = n^P c_L - \beta q^2 \qquad (1')$$

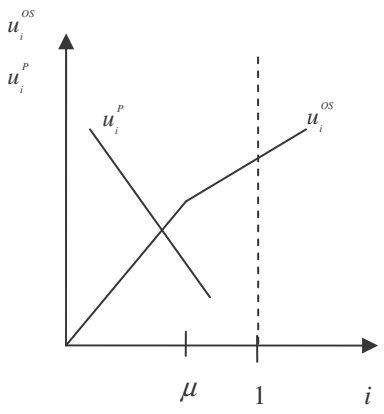
$$u_i^p = \alpha_1 n^p + dq - c_L \qquad (2')$$

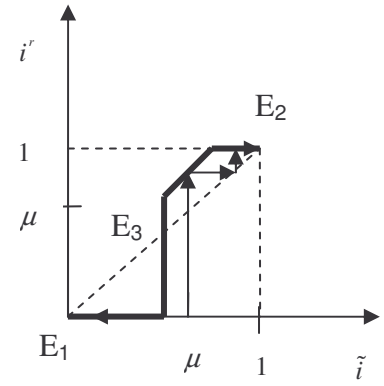
$$u_i^{OS} = \alpha_1 n^{OS} + \alpha_2 n_{OS}^D - c_i^A \text{ with } c_i^A = \begin{cases} 0 \text{ when i} \in [0, \mu] \text{ (developers)} \\ c_i^{EU} = \frac{(i-\mu)}{(1-\mu)} \overline{c}^{EU} \text{ when i} \in [\mu, 1] \text{ (end users)} \end{cases} \qquad (3')$$

Let us also assume that $\overline{c}^{EU} \leq (\alpha_1 + \alpha_2 \mu)$. This assumption bounds the magnitude of the adoption costs. Put differently, the end-user that exhibits the largest adoption cost $\left(\overline{c}^{EU}\right)$ always obtains a non-negative utility when he adopts the OS software. This hypothesis seems reasonable since the reverse would mean that the OS is "by construct" totally unsuited for a class of end-users. It has a practical implication since it rules out the inaction behaviour.

The resolution of the second step sub game takes as given the values of c_L and q chosen by the firms at the first step of the game. The variables to be determined are n^P*, the Nash equilibrium proportion of agents adopting the proprietary software, $n_D^{OS}*$, the Nash equilibrium proportion of developers and $n^{OS}*$, the Nash equilibrium proportion of agents adopting the Open Source software. We will expose this second step sub-game graphically. The respective slopes of $f_1(.), f_2(.)$ and c_i^A thus define several outcomes. These outcomes define simultaneously the number of equilibria and the stability of these equilibria. We can identify the following cases (see the whole graphical derivation of Nash Equilibria in *Appendix 2*) :

In the first cases (Cases 1a-b-c below), the adoption cost incurred by each successive end users is outweighed by the externality (on compatibility) generated by that users. The table below describes the conditions for these cases to occur. In cases (1a) and (1b), there are three Nash Equilibria of which only the two extreme are stable. In these two equilibria, either the commercial or the OS software is adopted but the two software never coexist at equilibrium. This case exhibits a "winner takes all" situation The basin of attraction of each equilibria is defined by the intersection between the two curves: when agents' expectation about the adoption of the OS software are below a critical mass defined by the intersection, they expect the proprietary software to impose. On the contrary, if expectations are beyond that threshold, the OS software imposes. In Figure 1c, the utility generated by the commercial software is always higher to that of the OS one, hence the only equilibrium is when all users adopt the commercial software.

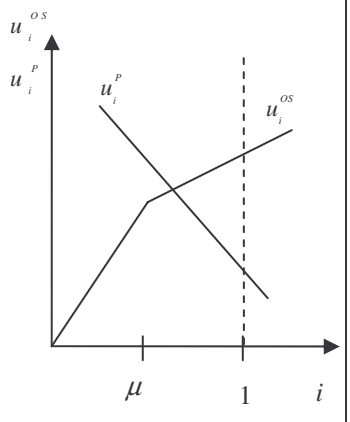


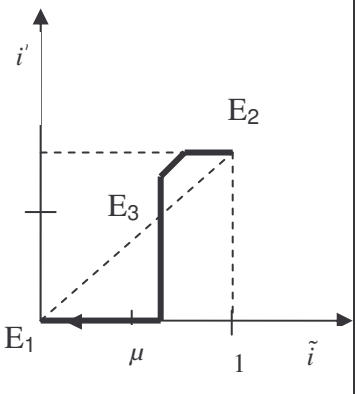


Case 1a

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) > \overline{c}^{EU} \\ dq - c_L \le \alpha_1(2\mu - 1) + \alpha_2\mu \end{cases}$$

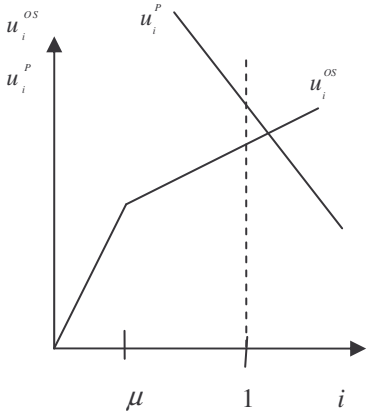


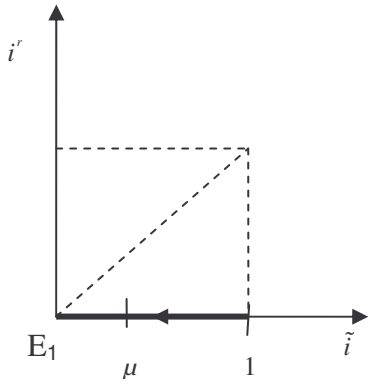


Case 1b

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) > \overline{c}^{EU} \\ dq - c_L \ge \alpha_1(2\mu - 1) + \alpha_2\mu \\ dq - c_L \le \alpha_1 + \alpha_2\mu - \overline{c}^{EU} \end{cases}$$



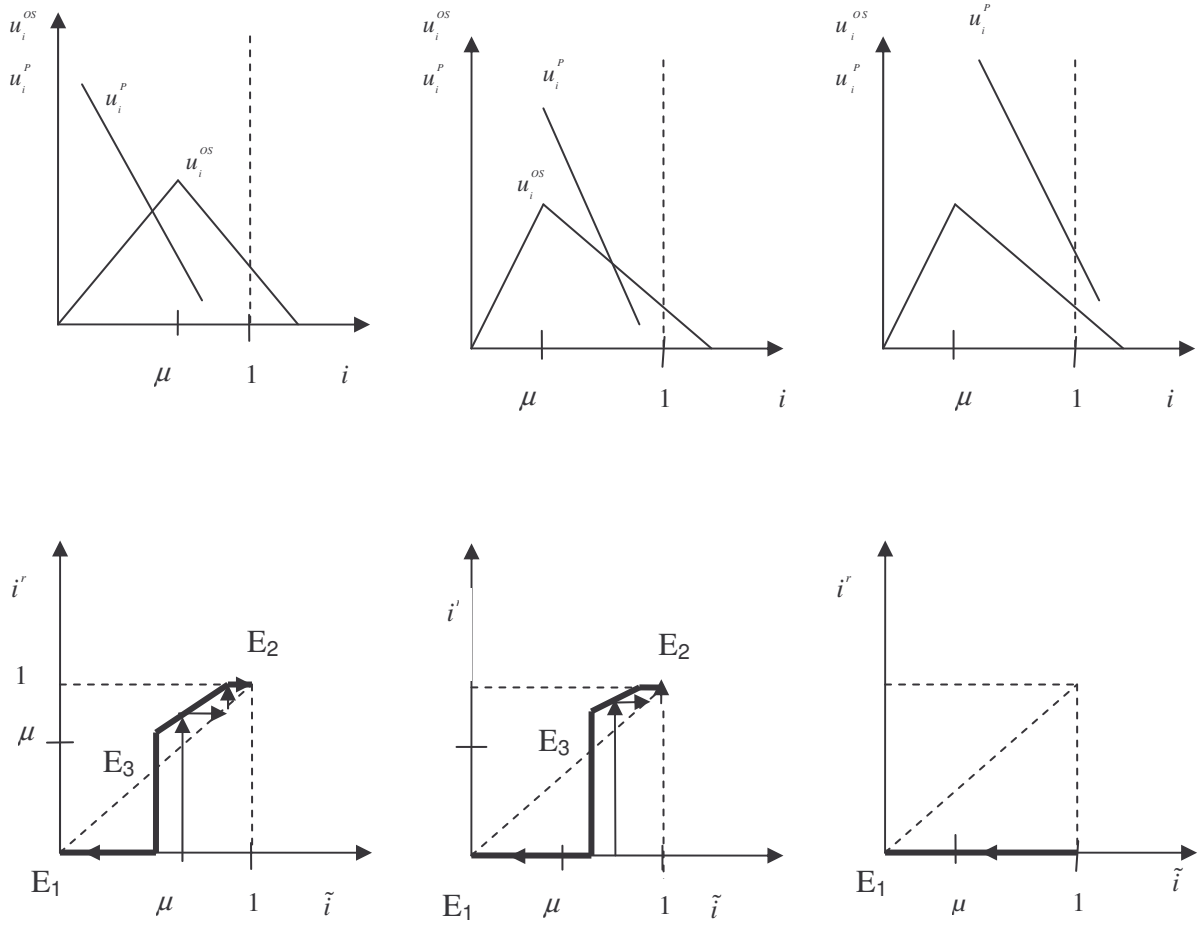


Case 1c

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) > \overline{c}^{EU} \\ dq - c_L > \alpha_1 + \alpha_2\mu - \overline{c}^{EU} \end{cases}$$

Without any prior knowledge about the magnitude of adoption costs, we also need to consider the case when the adoption cost of the OS software outweighs the benefit linked to its compatibility externality. This is depicted by the four remaining cases (depending of the respective slopes and the number of intersecting values).



Case 2a

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) < \overline{c}^{EU} \\ dq - c_L < \alpha_1(2\mu - 1) + \alpha_2\mu \\ dq - c_L < \alpha_1 + \alpha_2\mu - \overline{c}^{EU} \end{cases}$$

Case 2b

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) < \overline{c}^{EU} \\ dq - c_L > \alpha_1(2\mu - 1) + \alpha_2\mu \\ dq - c_L < \alpha_1 + \alpha_2\mu - \overline{c}^{EU} \end{cases}$$

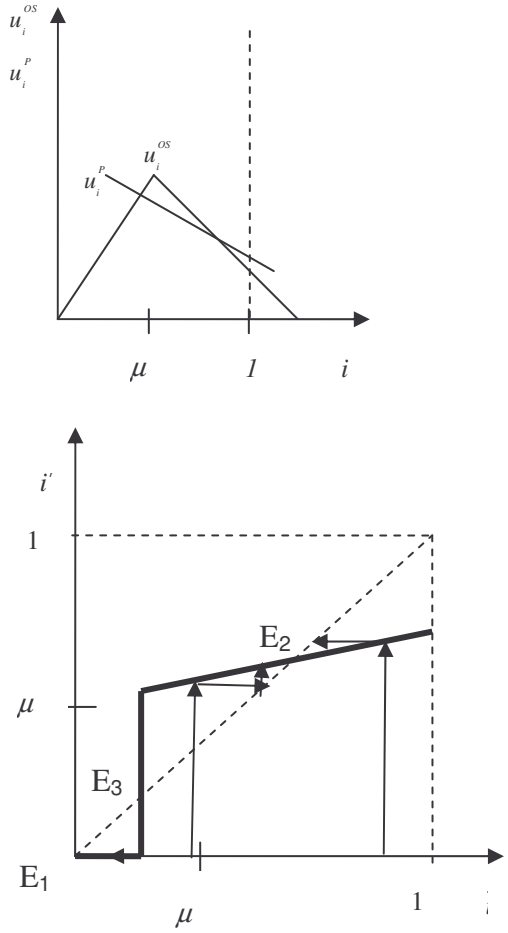
Case 2c

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) < \overline{c}^{EU} \\ dq - c_L > \alpha_1 + \alpha_2\mu - \overline{c}^{EU} \end{cases}$$

Cases 2a, 2b and 2c are analogous to Cases 1a, 1b, 1c respectively. In Cases 2a and 2b, there are again three equilibria where the intermediate equilibria (E_3) is unstable. Again, such situations may lead to a winner-takes-all competition.

Finally, Case 3 depicts a qualitatively new situation where the resulting equilibrium can be either the crowding out of the OS software or its *partial* adoption. In the last case, both commercial and OS software coexist at equilibrium. While developers all adopt the proprietary software, end users thus divide among the adoption of the commercial and OS software according to the magnitude of their adoption cost.



Case 3

Condition of appearance :

$$\begin{cases} \alpha_1(1-\mu) < \bar{c} \\ dq - c_L < \alpha_1(2\mu - 1) + \alpha_2\mu \\ dq - c_L > \alpha_1 + \alpha_2\mu - \bar{c} \end{cases}$$

We can summarize the equilibria of the second sub-game by the following table:

	Number of equilibria	Number of stable equilibria	Type of stable equilibria
Case 1a/2a	3	2	Proprietary Software only [P]
			OS Software only [OS]
Case 1b/2b	3	2	Proprietary Software only [P]
			OS Software only [OS]
Case 1c/2c	1	1	Proprietary Software only [P]
Case 3	3	2	Proprietary Software only [P]
			Coexistence of the two software[OS-P]

These equilibria stands for one given (q, c_L) strategy of the firm. Solving backward, we need then to establish the equilibria in the first sub-game.

3.2 The first step

3.2.1 General case

Once we have deduced the possible outcomes of the second step of the game for a given strategy of the commercial firm, we can go back to the first sub-game and determine the optimal strategy of the firm. At this stage, the firm has to choose the (q, c_L) couple maximizing its profit function. Given that the 2nd subgame equilibrium not always unique, the firm must consider that for each pair (q, c_L) it chooses at the first step, each of the possible issues associated at the second stage of the game. For the firm, such multiplicity implies that it has to consider the probability and the corresponding payoff of each equilibria. Put differently, the firm randomizes the equilibrium occurring at the second step by the probability for each equilibrium to occur. Consequently, we suppose that *i)* only stable issues have a positive probability to appear at the second step of the game[10]; *ii)* when there exist two stable issues, their respective probability are given by the size of their relative stability zone, as delimited by the position of the unstable equilibrium in the figures of section 32. Let us also suppose that users' initial expectations (prior beliefs) about the OS diffusion are equally likely (uniformly distributed). Without loss of generality, we suppose that firms are risk-neutral[11]. Let $n(q, c_L)$ be the probability that the equilibrium without OS users is reached when (q, c_L) are the actions of the firm at the first step of the game and $[1 - n(q, c_L)]$ the probability that the other stable (pooling or separating) equilibrium is reached. The expected profit of firms is given by;

$$\pi^e = n(q, c_L)\pi_1 + [1 - n(q, c_L)]\pi_3$$

where π_1 and π_3 are respectively the profits corresponding to the two stable issues of the second step game. The expressions of π_1 and π_3 are respectively given by

$$\pi_1 = c_L - g(q)$$

since in this case all agents adopt the commercial software. The expression of π_3 is slightly more complex and depends on the nature (pooling or separating) of the equilibrium to which π_3 corresponds. Its general expression is:

$$\pi_3 = \sup\left[0, 1 - i_2\right] c_L - g(q)$$

where i_2 is (if it exists) the rank of the first end-user (those being ranked according their increasing adoption costs) willing to choose the commercial software while the i_2 agents which adoption costs are lower than his adopt OS software; i_2 may as well be interpreted as the abscise coordinate of the second intersection of u_i^p and u_i^{OS} when this intersection exits, and when its coordinate is less than one (that explaining the $\sup[.]$ in the expression of π_3).

[10] The probability for an unstable equilibrium to occur is of null mass.
[11] Considering that they are risk-adverse only increases the formal complexity of the first step program, without changing its generic properties.

The value of $n(q, c_L)$ is deduced from the location of the second (unstable) equilibrium of the second step sub-game. More precisely, when this equilibrium exists, $n(q, c_L)$ is the abscise coordinate of the first intersection of u_i^p and u_i^{OS}. When this second equilibrium does not exists (and no more such an intersection between 0 and 1), $n(q, c_L)$ is equal to one. One can easily verify that as u_i^{OS} and u_i^{OS}, $n(q, c_L)$ is as a consequence a continuous function of the two action variables q and c_L. At last, qualification constrains apply. These are $q \geq 0$, $q \leq 1$ and $c_L \geq 0$.

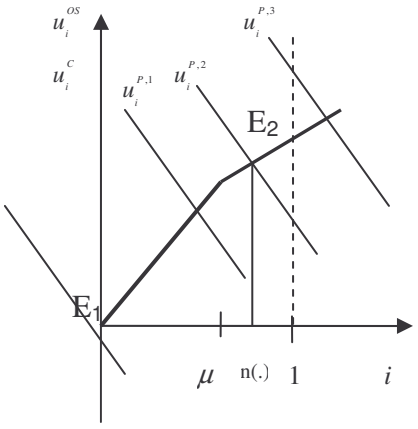
We then can prove the following proposition :

Proposition 3: An optimal outcome always exists for the firm at the first step of the game *(proof: see Annex 2)*

Corollary: An equilibrium of the two-step game always exists. When this equilibrium is not unique, equilibria correspond (except for cases of vanishing measure), to a single level of functionalities and price of the commercial software but to different levels of adoption of the commercial software and the OS software.

3.2.2 Linear-quadratic Specification

Let us illustrate this in cases 1a, 1b and 1c. In such cases, the curve depicting the OS software is always increasing and the slope of the curve depicting the commercial one is also identical. The only changing element between these three cases lies in the intersection between the U_P curve and the origin axis.



Considering the definition of the U_P curve, this intersection is directly ruled by the control variables of the firm q and c_L. Let us temporarily consider that q is constant. Obviously, the profit of the commercial firm increases when the U_P curve shifts left and when the firm increases its licence cost. In that sense, the firm can manipulate the position of this curve (by setting various quality levels and charging various licence costs) so as to maximize its profit. At equilibrium E_1, it could charge a "monopoly" price by exhausting consumers' surplus and its profit is maximum. However, the firm cannot directly rule the effects linked to the development and compatibility externalities. For that reason, it risks to be crowded out anytime the initial population of OS developers is higher than $n(q, c_L)$ (defined by the third equilibrium). The firm risks to be crowded out with a probability $(1 - n_0)$. Considering a risk neutral firm, we then need to write Firm's expected profit as:

$$\pi^e = n(q, c_L)\left(n^P c_L - \beta q^2\right) + (1 - n(q, c_L))\left(-\beta q^2\right) = n(q, c_L)\left(n^P c_L\right) - \beta q^2$$

In a first analysis, we can illustrate the previous arguments by some numerical examples. Let us consider for instance a case typical of "Case 1" ($\alpha_1 = 5, \alpha_3 = 5, \beta = 1, \mu = 0.9, d = 5, \overline{c}^{EU} = 0.2$), when the compatibility externality (depicted in the linear case by Coefficient α_1) outweighs the adoption cost incurred by the marginal user. We can plot the expected profit of the firm for different (q, c_L) strategies[12].

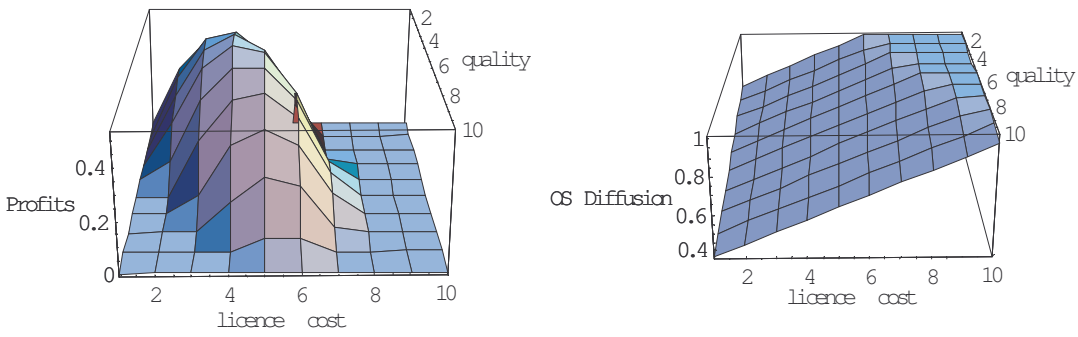


Fig. 4: Firm's expected profit (left) and OS expected diffusion (right) $\left(\alpha_1 = 5, \alpha_3 = 5, \beta = 2, \mu = 0.9, d = 5, \overline{c}^{EU} = 0.2\right)$; $\pi^{max} = 0.52 > 0$ reached for $\left(c_L^* = 3 \text{ and } q^* = 0.2\right)$.

As one can see in the previous figure, the maximum expected profit $\left(\pi^{\max} = 0.52 > 0\right)$ as reached for an interior value of the strategy set $\left(c_L^* = 2 \text{ and } q^* = 0.2\right)$. The right figure shows that the commercial firm can indirectly (and partially) control the diffusion of the OS software by setting different (price,quality) strategies. However, we see in this example, that because of the development cost, the firm has no incentive to crowd out the OS software completely. Instead, the *expected* diffusion of the OS software corresponding to firm's optimal choice is equal to 0.7. As it has already been pointed out, the firm faces a trade-off: either it selects a strategy that crowd out the OS project by setting a high quality and/or low licence cost, or it has the "inverse" strategy (low quality and/or high price). In the last case, it obtains a maximal profit when the "P" equilibrium (no OS adoption) is selected but faces the risk of a massive adoption of the OS project and of being itself crowded out. In the first case, this risk vanishes but the profit obtained when the "P" equilibrium decreases. In the case presented, the optimal behaviour of the firm has to be comprehended as a balance between these two strategies. We could also mitigate this result by taking into account firm's risk aversion. However, one could bet that such a change would not change qualitatively our results. Intuitively, a more risk-adverse firm would likely minimize the risk of being crowded out so that the optimal strategy would be more conservative (quality increase and/or price decrease).

The following figures exhibit another interesting case:

[12] For each strategy, we consider the relative positions of the U_P and U_{OS} curves. From this comparison, we deduce the case (1a-1b-1c) and compute the expected profit and the expected diffusion of the OS project accordingly. If $U_P < 0\left(\forall i \in [0,1]\right)$, no users is willing to adopt the commercial software. Knowing that, the firm does not make any investment on quality. Hence firm's profit is null is that case.

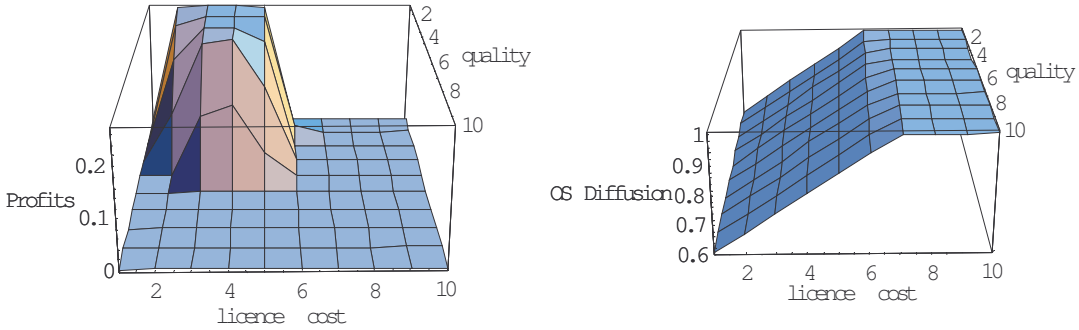


Fig. 5: Firm's expected profit (left) and OS expected diffusion (right) $\left(\alpha_1 = 5, \alpha_3 = 5, \beta = 2, \mu = 0.9, d = 1, \overline{c}^{EU} = 0.2\right)$; $\pi^{max} = 0.4 > 0$ with for $\left(c_L^* = 3 \text{ and } q^* = 0.2\right)$; expected OS diffusion = 0.6.

To keep things comparable, all parameters but coefficient d are unchanged. We lowered this coefficient (now equal to 1 instead on 5 in the previous case). That means that users attach less importance to specific functionalities. Taking this element into account, the optimal strategy of the firm is to concentrate on generic functionalities and not to invest in specialized ones $(q = 0)$. Again, one interesting feature is that the firm has still no interest in crowding out the OS software.

Let us consider a third case. Again, all parameters but d, are kept constant. Now, $d = 15$. That means that specialized functionalities are now more important than generic ones in users' utilities.

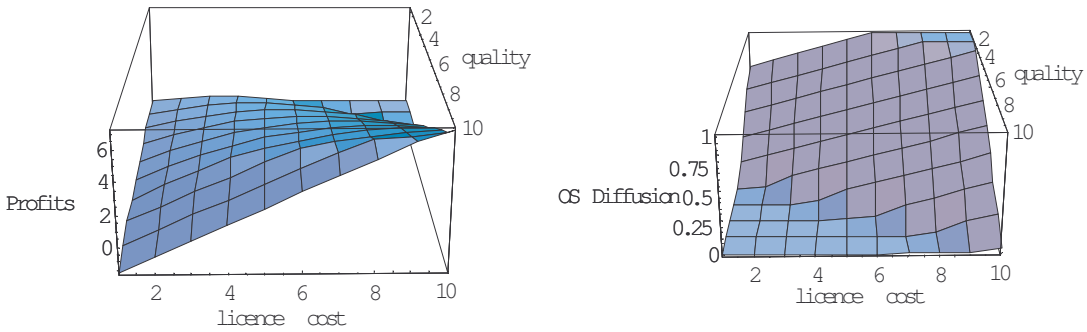


Fig. 6: Firm's expected profit (left) and OS expected diffusion (right) $\left(\alpha_1 = 5, \alpha_3 = 5, \beta = 1, \mu = 0.9, d = 15, \overline{c}^{EU} = 0.2\right)$; $\pi^{max} = 7.04 > 0$ with $\left(c_L^* = 9 \text{ and } q^* = 1\right)$.

We now observe that firm's optimal strategy is here to develop all specialized functionalities. This is a quite natural result since these functionalities are here determinant in users' adoption decisions. Let us interpret this case in more details. We observe that the profit maximizing strategy is reached when the firm intensifies its R&D effort. ($c_L = 9$, $q = 1$). At this strategy, the expected diffusion of the OS software is nearly null (equal to 0.03%). If we compare the utilities of the two software, we see that the development effort (depicted by $\alpha_3 = 5$) is much lower than that of the firm (depicted by $d = 15$). Hence, the firm has an incentive to invest on quality in order to differentiate its software from the OS software. Doing so, the probability of diffusion of the OS is degraded. When the R&D development effort of the firm is maximal ($q = 1$), the OS software is crowded out. Apparently, this situation ends up as a monopoly. However, this situation is different from the one prevailing with no OS software: the OS software act as a potential incumbent. Even if it cannot compete with the commercial firm (due to the asymmetry on development efforts i.e. $\alpha_3 << d$), the sole presence of the OS software makes the firm invest more on software quality than what it would have done without the OS software.

Indeed, in the absence of any competitor (be it another commercial firm or an OS project), the firm would have acted as a monopolist: its R&D effort would have been minimal (it would have proposed the least quality i.e. $q = 0$) and it would have set a monopoly price (exhausting users' surplus). Instead, the threat of the OS software makes it invest on quality. Hence, the software proposed by the firm is of higher quality which benefit to all users.

4 Further research

This paper is a first attempt to depict the competition and coordination issues arising from the existence of OS projects: in a framework where externalities favour lock-in and winner-takes-all situations, the competition between OS and commercial software have been analyzed using a two step game: in a first step, the firm sets the quality and the price of its software. In a second step, end users and developers react by adopting the commercial software or engaging into the OS project. We have shown that such situation often lead to multiple equilibria. Because the commercial firm can influence but cannot entirely select the most favourable equilibrium, it faces a trade-off: either it adopts a "low price – high quality" strategy and deter the OS project; or it adopts a "high price – low quality" strategy and bears the risk of being crowded out by the OS software. We show that the strategy of the commercial firm can be understood as a balance between these two basic strategies. Depending on the preferences of the users, we pointed out some cases where *i)* the firm has no interest to invest in quality or on *ii)* on the contrary, it has an incentive to develop all functionalities to maximize its profit. These first findings are rather and now need to be enriched in two directions. First, we need to investigate the welfare issue: intuitively, the presence of the OS project can be interpreted as a "threat" (incumbent firm) and as such, stimulates the commercial firm to decrease its licence costs or to improve quality. This has a global positive impact on users. However, such intuition has to be confirmed by a more precise analysis and by taking into account users' heterogeneity regarding the OS project. Second, we have developed a general approach that needs now to be calibrated to some specific contexts: Linux versus MS Windows, OpenOffice versus MS Office, PSP versus ASP, Apache, *etc*.

References

Bonaccorsi, A., and C. Rossi (2003): "Why Open Source Software Can Succeed," *Research Policy*, 32, 1243-1258.

Cremer , J., and A. Gaudeul ((2004)): "Some Economics of the Open-Source Software," *Réseaux*, (forthcoming).

Dalle, J.-M., and N. Jullien (2003): "'Libre' Software: Turning Fads into Institutions?," *Research Policy*, 32, 1-11.

Franke, N., and E. Von Hippel (2003): "Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software," *Research Policy*, 32, 1199-1215.

Gaudeul , A. (2003a): "The (La)Tex Project: A Case Study of Open Source Software."

— (2003b): "Open Source Software Development Patterns and License Terms."

Hann , I.-H., J. Ropers , S. Slaughter , and R. Fieding (2002): "Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache Http Server Project," GREMAQ-IDEI 2002 Conference on Software and Media Industry, Toulouse,

Lerner, J., and J. Tirole (2001): "The Open Source Movement: Key Research Questions," *European Economic Review*, 45, 819-826.

— (2002): "The Scope of Open Source Licensing."

Muselli , L. (2004): "La Licence Informatique: Un Outil Stratégique Puissant Pour Les Éditeurs De Logiciels ?," Second Workshop of the Research Consortium "ICT & Society", Marne-la-Vallée, France, january 2004 28-29th.

Mustonen, M. (2003): "Copyleft--the Economics of Linux and Other Open Source Software," *Information Economics and Policy*, 15, 99-121.

Mustonen, M., and M. Leppämäki (2003): "Spence Revisited - Signalling with Externality: The Case of Open Source Programming," 5th Berlecon Conference on the Economics of I.T., Berlin, June, 13-14th.

Schmidt, K., and M. Schnitzer (2003): "Public Subsidies for Open Source ? Some Economic Policy Issues of the Software Market," *Harvard Journal of Law and Technology*, 16.

Shapiro, C., and H. R. Varian (1999): *Information rules: A strategic guide to the network economy*, x, 352.

Von Krogh, G., and E. Von Hippel (2003): "Special Issue on Open Source Software Development: Editorial," *Research Policy*, 32, 1149-1157.

West, J. (2003): "How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," *Research Policy*, 32, 1259-1285.

Zimmermann , J.-B., and N. Jullien (2002): "Le Logiciel Libre : Une Nouvelle Approche De La Propriété Intellectuelle," *Revue d'Economie Industrielle*, 99.

Appendix 1: Nash equilibria of the Step 2 sub-game

Proof of Proposition 1: ***c***onsider the choice made by each agent *i* when he / she observes the pair (q, c_L) at the second step of the game:

- Choosing *P* if $f_1(\tilde{n}^P) + dq - c_L \geq \sup\{0, f_1(\tilde{n}^{OS}) + f_2(\tilde{n}_D^{OS}) - c_i^A\}$
- Choosing *OS* if $f_1(\tilde{n}^{OS}) + f_2(\tilde{n}_D^{OS}) - c_i^A \geq \sup\{0, f_1(\tilde{n}^P) + dq - c_L\}$
- Choosing $\varnothing$ if $0 \geq \sup\{f_1(\tilde{n}^P) + dq - c_L, f_1(\tilde{n}^{OS}) + f_2(\tilde{n}_D^{OS}) - c_i^A\}$

Recall that $\tilde{n}_D^{OS}$ is associated to $\tilde{n}^{OS}$ in such a way that, by definition, $\tilde{n}_D^{OS} = \inf[\tilde{n}^{OS}, \mu]$. The three functions contributing to the individual choice are continuous relatively to their arguments $\tilde{n}^P$ and $\tilde{n}^{OS}$. If n^P and n^{OS} represent respectively the size of the total proportions of users choosing *P* and *OS* as the expected individual choices are $\tilde{n}^P$ and $\tilde{n}^{OS}$, then n^P and n^{OS} vary also continuously with $\tilde{n}^P$ and $\tilde{n}^{OS}$. Let us now consider the transformation from $([0,1],[0,1])$ to $([0,1],[0,1])$ associating to each pair $\{\tilde{n}^P, \tilde{n}^{OS}\}$ the pair $\{n^P, n^{OS}\}$. If they exist, the fixed points of this transformation are also the Nash equilibria of the second Step sub-game since in this case and for each agent, the best answer to the expected magnitudes obtained by aggregating the individual choices lead to confirm the individual optimality of these choices. Moreover, the transformation is continuous and defined from a compact subset on itself. It then admits at least a fixed point on this compact subset $([0,1],[0,1])$, this (these) fixed point(s) being Nash equilibrium (equilibria) of the sub-game. Suppose that $\{n^P*, n^{OS}*\}$ is a (one of the) fixed point(s) of this transformation. The size of the population of developers adopting the Open Source software $n_D^{OS}*$ can then be deduced elementarily from the relation of definition $n_D^{OS}* = \inf[n^{OS}*, \mu] = \tilde{n}_D^{OS} = \inf[\tilde{n}^{OS}, \mu]$, that completes the demonstration. ■

Proof of Proposition 2: let us denote n^{P+}, $(0 \leq n^{P+} \leq 1)$, the proportion of commercial software (*P*) users sufficient to provide a positive utility to each *P* user by the way of the positive externalities they generate on each user, i.e. such that $f_1(n^{P+}) + dq^+ - c_L{}^+ > 0$. Suppose that $\{\tilde{n}^P, \tilde{n}_D^{OS}, \tilde{n}^{OS}\} = \{1,0,0\}$. The utility derived for each agent from the commercial software is then given by $u^P = f_1(1) + dq^+ - c_L{}^+ \geq f_1(n^{P+}) + dq^+ - c_L{}^+ > 0$. In the same time, the utility provided by the *OS* software is $u_i^{OS} = f_1(0) + f_2(0) - c_A^i \leq 0, \forall i.$ Whatever the agent *i* considered, the inequality $f_1(\tilde{n}^{P+}) + dq^+ - c_L{}^+ \geq \sup\{0, f_1(0) + f_2(0) - c_A^i\}$ then holds.

By transitivity, $u_i^{OS} = f_1(0) + f_2(0) - c_A^i \leq \sup\{0, f_1(0) + f_2(0) - c_A^i\}$ is verified $\forall i$ and *P* is the best answer for all end users and developers. As a consequence, $\{n^P, n_D^{OS}, n^{OS}\} = \{\tilde{n}^P, \tilde{n}_D^{OS}, \tilde{n}^{OS}\} = \{1,0,0\}$. That demonstrates *Proposition 2.* ■

Derivation of the Nash Equilibria: for each set of values of the parameters and of the variables c_L and q, we consider simultaneously two related figures: on the upper chart, we display the utility of the i^{th} agent (marginal) when he adopts the commercial or the OS software; on the bottom chart, we deduce the proportion of agents choosing OS, according to the expected level of OS users.

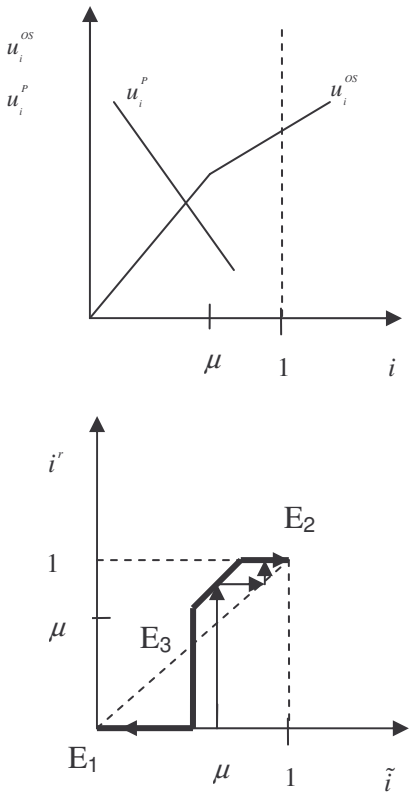


Figure A1 (a-b) : Equilibria of the second sub game

On *Figure A-1a*, the horizontal axis identifies the location of the i^{th} agent. On the ordinate axis, we represent its level of utility when he chooses the commercial or the OS software, while all agents before him have chosen OS. The equation of the curve depicting the commercial software is $\left(\alpha_1(1-i)+dq-c_L\right)$. That of the OS software is $\left(\alpha_1 i+\alpha_3 i\right)$ for $i\in[0,\mu]$ and $\left(\alpha_1 i+\alpha_3\mu-\frac{1-i}{1-\mu}\overline{c}^{EU}\right)$ for $i\in[\mu,1]$. The slope of the u_i^p-curve does not depend on the type of agent (developer or end-user) examined but only of the level of externalities generated by the commercial software, that explains the linearity of u_i^p. According to Equations 1'-2'-3', the slope of u_i^{OS} depends conversely on the type of agent : as $i\in[0,\mu]$, the slope of the curve is given by $\left(\alpha_1+\alpha_2\right)$ and as $i\in[\mu,1]$, this slope is given by $\left(\alpha_1-\frac{\tilde{c}^{EU}}{1-\mu}\right)$. According to the sign of this last term, we are able to distinguish two series of cases: *i)* cases where the adoption costs only dampen the effect of adoption externalities and *ii)* cases where the opposite relation is verified (adoption costs are higher than adoption externalities for the end-user). The curves u_i^p and u_i^{OS} may exhibit no, one or two intersections within the interval $[0,1]$.

Let us consider first the case of a single intersection. In that case, the abscise coordinate of the intersection corresponds to the critical level of adoption of OS sufficient to provoke the adoption of the OS. Supposing that all agents $j\in[0,i]$ adopt the OS, this intersection indicates that the i^{th} is then indifferent between adopting the OS or the commercial software and the subsequent agents will prefer the commercial software. Such cases cannot be excluded when the adoption costs are dissuasive for some end-users.

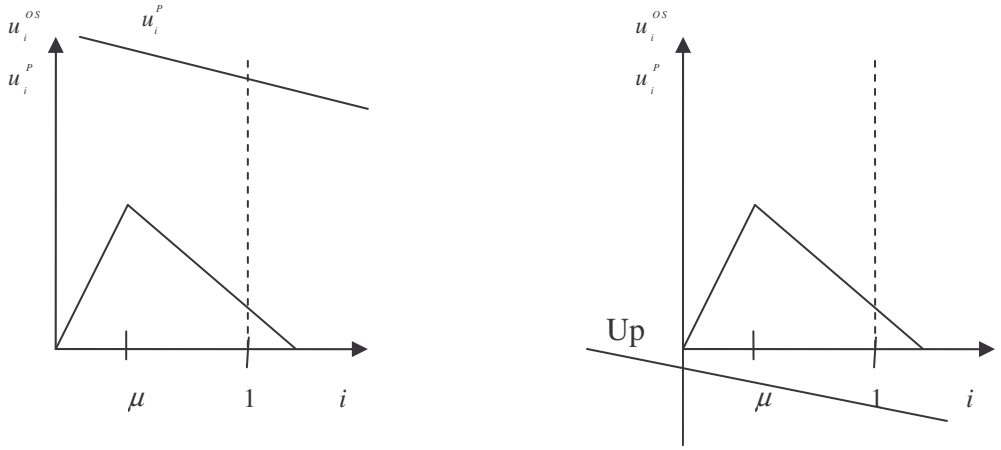
On *Figure A-1b*, the abscise coordinate corresponds to the expected size of the population of OS users. This *expected* size is considered by agents when they take their adoption decisions. The ordinate depicts the proportion of users of the population choosing to adopt OS according to the expectations expressed in the abscise coordinate. The relation represented is then the relation between the expected and effective levels of OS population. Agents are supposed to adopt the OS in the reverse order of their adoption costs (end users endowed with the lowest adoption costs, adopt firs). Thus, the intersections of this function with the 45° curve can be

regarded as levels the OS population such that if they are expected by the population, they are also confirmed by their rational choices; this kind of states have the properties of symmetric Nash equilibria.

Figures A1-a and *A1-b* can be related in a simple way: as long as the i^{th} agent does not hope adopting OS when agents $j \in [0,i]$ are expected to have adopted it, the function associating the desired level of adoption with the expected level one remains below the 45° curve. The 45° curve is cut up on *figure 1b* (in some cases with a vertical intersection), when the (first) intersection of u_i^p and u_i^{OS} is attained on *figure 1a* in an increasing part of function u_i^{OS}. This intersection defines a separating Nash equilibrium, in the sense that the agents with low adoption costs adopt OS while the others use the commercial software. When, after such intersection, the slope of the function u_i^{OS} is still positive after $i = \mu$, there exists, for a certain value of *i*, a second level of OS users such that if the size of OS users is expected by agents greater or equal to this level, all agents choose to adopt OS. In this case (where the second equilibrium of section 3.2. rules as well), the curve connecting the OS adoption level to the expected OS adoption level has the form depicted in *figure 1b*. There is a Nash equilibrium at the NE corner of the box. At last, when no agent is supposed to adopt OS, the equilibrium of Proposition 2 is at the SW corner of the box.

The local stability of the Nash equilibria can also be analysed in the case of static expectations with the help of *figure A-1b*. Let us consider the case 1a with three equilibria. Suppose thus that agents expect that the size of the OS users is somewhere between the two first equilibria. Then, their answer will be for all to adopt the commercial software. We conclude that all initial expectation of a relatively low level of OS users rapidly drives to the Nash equilibrium where the OS software is fully crowded out. Suppose now that the initial expected size of the OS users is somewhere between the second and the third equilibria. Then, agents choosing to adopt OS are more numerous than expected; they correct their expectations, the number of OS users increases by now… until the third equilibrium had been reached where only the OS software remains.

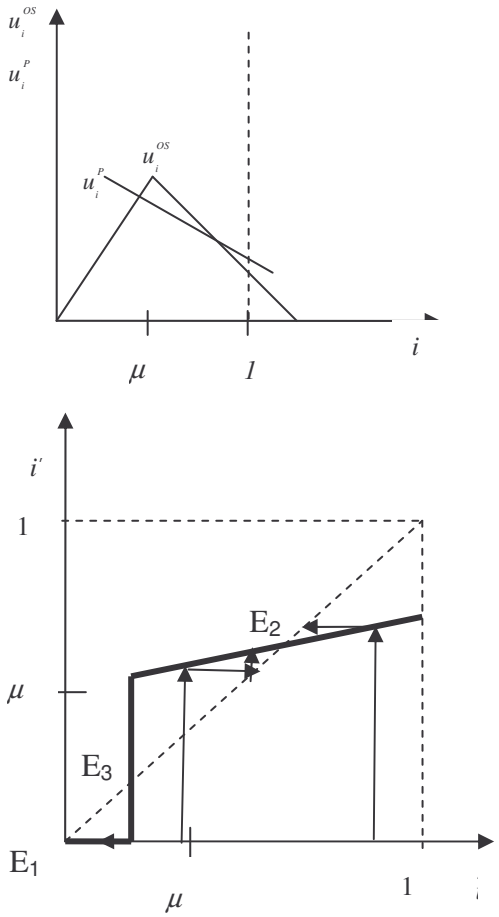
If there is no intersection, one of the two following cases is prevailing:



It is straightforward to derive the Nash Equilibria in such cases. On the left figure, the utility of the commercial software is always higher to that of the OS. The only Nash Equilibrium is thus an equilibrium in which all users adopt the commercial software. On the right figure, the utility of the commercial software is inferior to that of the OS one, for any user *i*. Hence no user is willing to adopt the commercial software and the only Nash equilibrium is when all users adopt the OS software.

The last case is when there are two intersections. If there are two intersections, one intersection is necessarily located on the first segment $([0,\mu])$ and the other on the second segment $([\mu,1])$. In that case, the second segment of u_i^{OS} is necessarily decreasing[13]. Using the same reasoning, we can infer that there are three equilibria in that case while the "upper" equilibria is mixed (coexistence of OS and commercial software).

[13] If increasing, there could just be one intersect since all curves are strictly linear.



The same stability analysis can be reproduced in all cases and figures. When there exists only one equilibrium, this equilibrium is stable; when there are three equilibria, the extreme are stable (even when one of them is separating) while the intermediate (always separating) is unstable.

Appendix 2: Nash equilibria of the first sub-game

Proof of Proposition 3. Consider the function to be maximized $\pi^e = n(q, c_L)\pi_1 + [1 - n(q, c_L)]\pi_3$ and the constraints and restrictions $\pi_1 = c_L - \beta q^2$, $\pi_3 = \sup\left[0, 1 - i_2\right]c_L - \beta q^2$, $q \geq 0, q \leq 1$, $c_L \geq 0$. When $f_1(.)$ and $f_2(.)$ are continuous and c_A^i defined as in the general case, i_2 and when $n(q, c_L)$ varies continuously with q and c_L, so does π^e. The value of q is by definition bounded from below and above. The value of c_L is bounded from below only. Suppose then that for a given value of q, the best choice for the firm would be to increase indefinitely the price of the commercial software. Consider the answer of the consumer *i,* given by $u_i^p = f_1(n^p) + dq - c_L$. Even with a maximal level of externalities and given that $f_1(1) = k_1, (0 < k_1 < \infty)$, the value of u_i^p can not definitively remain positive as long as c_L increases. Then, whatever the value of q considered, it exists a value of c_L from which the commercial software cannot be adopted, even when externalities are maximal. Let us call it $c_L^{MAX}(q)$.Then, the maximum of the problem is defined on the subsets $q \in [0, 1], c_L \in \left[0, c_L^{MAX}(1)\right]$. This subset is compact and thus insure the existence of a maximum for the objective function $\pi^e = n(q, c_L)\pi_1 + [1 - n(q, c_L)]\pi_3$. ■