Code Development: Open vs. Proprietary Source*

(Preliminary - Comments welcome)

German Daniel Lambardi†

March 10, 2008

Abstract

We develop a model in which a software is obtained from an initial amount of code after successfully overcoming a sequence of steps. The owner of the initial code must decide between carrying out these steps under an open or a proprietary source environment. Open source development will allow the initial code owner to save on developing cost aided by a community of "sophisticated" user-developers, however it will imply lower future income on "unsophisticated" end-users. With this model we try to understand why some profit seeking firms may donate code and start open source projects. The dynamic structure of sequential code improvement will provide an alternative explanation on user - developers collaboration. Finally we introduce competition with an existing proprietary source alternative. We show that the incumbent might find optimal to lower prices to user-developers to reduce the aid provided by the community and therefore deter entry or prevent the development of the alternative as Open Source.

*I am endebted to Jacques Crémer for his advice and guidance. I am grateful to Guido Friebel and Yossi Spiegel for early observations. I would also like to thank the participants of the 2nd ICT Telecom Paris conference, the OSSEMP workshop at Limerick Ireland 2007, the First International Workshop FLOSS Nice - Sophia Antipolis 2007 and the discussants Yann Ménière and Maria Laura Parisi. I am solely responsible for any mistakes and errors.

†GREMAQ, Université de Toulouse glambardi@gmail.com

1 Introduction

The Open Source Software phenomena (hereafter denoted OS) has attracted considerable attention from the economists' community in the last few years. Several OS projects became successful alternatives to Proprietary Source Software (hereafter denoted PS) and despite the heterogeneity of success it is now consider as a feasible way to develop code and produce software. Moreover, judging by some recent field studies (i.e.: Dahlander and Magnusson (2005), West and O'Mahony (2005), Bonaccorsi et. al (2004)) OS development besides being feasible might also be a profitable business model for firms.

Although the OS phenomena is quite complex, a broad and useful taxonomy has been proposed by West an O'Mahony (2005) between "community-initiated projects" and "spinout projects". The first category is the most familiar model of OS development. Such projects are usually initiated by one or more individuals that don't share a common employer. This projects tends to remain community managed and although they might allow firms contributions they are structured to prevent firm takeover. Examples of this are Linux or Apache software. In the second category, also called "sponsored", usually a for-profit firm (the sponsor) starts a OS project normally by releasing some valuable internally developed code and inviting an external community to join and collaborate with the code development. Along with the "sponsor" developing efforts, an important amount of development help is obtained from users if the project is successful. This kind of projects tend to remain sponsor-controlled[1] and firms usually make profits on complementary software or services. Some examples in this category are MySQL, JBoss or IBM's Eclipse.

In this paper we will focus our attention to the "spinout" kind of projects. A motivating example of the "spinout/sponsored" project is JBoss. This OS software is an application server system based on the Java 2 Enterprise Edition (J2EE). Marc Fleury, founder of the for-profit company JBoss Group, released the first version of the OS JBoss in 1999. By 2003 the company was successful in developing a community around the product and already employed 30 full time developers. By that time, JBoss started to grow in popularity (over

[1] An important exemption is Mozilla.

two million downloads[2]) at the expense of IBM and BEA existing proprietary alternatives. Since JBoss software is free of charge, profits comes from providing support and consulting. As Fleury stated in an interview[3] "we just use open source as R&D and recruitment...". The company was bought by RedHat in 2006 in $350 million.

Our main interest in this paper is to provide a model to understand and explore the "spinout" OS development in contrast to the traditional Proprietary development. The way we choose to approach this issue is to model a dynamic multistage process in which the code is developed. The model assumes that a software is obtained from an initial amount of code, after successfully overcoming a fixed sequence of stages. The probability that the code advances from one stage to the next one is assumed to depend on the amount of individuals devoted to the project at each stage. The owner of the initial code, a profit maximizing firm, must decide between carrying out these steps under an OS or a PS environment. Although we initially assume that this firm faces no competition from an alternative software, we later extend the model to capture this aspect.

Open source development will allow the firm to save on programming costs aided by the free programming time provided by a community of "sophisticated" user-developers (hereafter denoted UD). These savings come at a cost, since the royalty free distribution of OS will not allow the firm to make money by selling licences once the software is obtained. We will assume however that OS returns are not zero. OS firms are usually able to sell complementary proprietary software as well as support and customization services to unsophisticated end-users (hereafter denoted EU). [4]

The PS development, on the contrary, faces higher development costs since the code owner must pay for each individual working on the progress of the software. On the other hand the firm will be able to charge money once the software is obtained providing a higher future income than the OS alternative.

The set up considers that UDs are able to profit from the raw code at each stage. They

[2]http://www.serverwatch.com/sreviews/article.php/2190151

[3]http://www.news.com/2008-1082-994819.html

[4]see Dahlander and Magnusson (2005)

derive utility from the use of the code but they must spend time programming to adapt and make usable the code for their purposes. From the UD perspective each time the code advances to the following step, it becomes more valuable (i.e.: they derive more utility) and also it requires less programming time to be "usable". In a small extension of the model we show that the dynamic structure of the sequential code improvement will also help us to understand the dynamic of cooperation: UD will collaborate with the OS firm because if their contributions are included in the next version of the code, then they will save programming time in the next stage.

With this model we explore when it is more likely that a for-profit firm starts an OS projects. Some results can be highlighted: a) If the firm decides to develop the code as OS, the overall probability to overcome all the stages to get the final software is smaller. This might hurt the welfare of EU and represents a threat that should be taken in consideration when doing welfare analysis of the OS phenomena. b) UD might find optimal to collaborate in an OS project at early stages even if at that point they obtain negative utility in order to spend less time in the future adapting the code. c) When competition is introduced in the model and an existing alternative software is able to price discriminate, the incumbent might find profitable to lower the price charged to UD to deter entry or to avoid OS development, by reducing the size of the community that cooperates with the development.

2 Basic Framework

The following model is inspired in Aghion, Dewatripont and Stein (2005) framework of academic vs. private-sector research. Here a firm E owns a certain amount of code C_0 that can be used to produce a software S. To obtain S, the code has to increase and undergo certain transformations. This transformations are modeled as a finite number k of stages which the code must go through. At the final stage S is obtained. The overcoming of each stage requires individuals (programmers) working on the development of the code, however this development is stochastic and the probability that at any stage j the code C_j advances

one step towards C_{j+1} is given by:

$$\Phi(N_j) = 1 - (1-q)^{N_j}$$

where N_j is the amount of individuals working on the code development and q is the probability that the code advances one step ahead (success) if only one individual is working on the code. The function $\Phi(N_j)$ is just the probability of at least one success among a group of N_j individuals working on the code.

This development of the code can be done in an open or proprietary environment. If the firm choose to make the code open, some users will be able to read and understand the code and therefore cooperate with the development. By the same token the open environment reduces the ability of the firm to make profits on S due to the royalty free distribution.

The firm faces a population of potential code users. This population is composed by two categories of individuals: "sophisticated" User Developers (UD) and "unsophisticated" End Users (EU). If the firm makes the code C_j open at stage j, the UDs are able to profit from it: they can use C_j to perform some private activity for which they derive utility but they need to spend time programming to be able to use the raw code. The EU group on the contrary can only profit from the code if the software is obtained in the last stage.[5]

To be more specific, the utility a UD derives from the finished software S is S. If C_j is available at stage j, the **gross** utility per period a UD derives from using it is $\kappa_j S$, with $0 < \kappa_j < 1$ and $\kappa_{j+1} > \kappa_j > \kappa_{j-1}$. The parameter κ_j captures the idea that C_j is not a "finished" software: features could be missing and performance could be poor. To be able to use the raw code, each UD must spend per period λ_j units of time programming, with $\lambda_j > 0$ and $\lambda_{j+1} < \lambda_j < \lambda_{j-1}$. We can interpret λ_j as a use and maintenance cost: while using C_j an important amount "bugs" could show up that requires reprogramming fractions of C_j, compatibility issues might appear with other software the UD is using that should be solved. Also the use C_j would require new essential features to be developed.

[5]Through out the papers when we refer to UDs we will talk about "individuals" with IT capabilities. However they could as well be other firms with IT capabilities that use the code for their purposes.

As the software progresses, features are added and problems are solved, so the software becomes more valuable ($\kappa_j \to 1$) and less costly to use ($\lambda_j \to 0$). The modifications and changes each UD performs to use the code represent potential improvements that can make C_j advance to C_{j+1} if they are shared with the rest of the community. For the moment we will assume that UDs are willing to cooperate by sharing their code modifications and they face no cost in doing so. A "reduced form" to model the UD collaboration at stage j is to say that from firm's E perspective it receives free help from an amount m_j of individuals working on the code development.

The desutility a UD derives from the time spent programming is l, therefore the **net** utility per period of using the code at stage $j < k$ is

$$U_j = \kappa_j S - \lambda_j l$$

This expression can be transformed without loss of generality to

$$U_j = S - \alpha_j l$$

with $\alpha_j = \frac{\lambda_j}{\kappa_j}$ and $\alpha_{j+1} < \alpha_j < \alpha_{j-1}$. At the final stage k the UD enjoys a "finished" software so $\kappa_k = 1$. For simplicity we also assume $\lambda_k = 0$[6].The utility per period derived from the finished software S is then S. Therefore:

$$U_j \begin{cases} S - \alpha_j l & \text{if use code at } j < k \\ S & \text{if use code at } j = k \end{cases}$$

We further assume that UD are heterogeneous in the desutility l (i.e. they have different IT capabilities)[7]. The total amount of UDs is given by M, and the fraction of them that has l lower than a certain $\widehat{l}$ is $G(\widehat{l})$. For simplicity in the rest of the paper we will consider only $l \geq 0$, notice however that this formulations could also include individuals intrinsically

[6]However it can be said that usually even a "finished" sofware has problems and demands a lot of time.

[7]At the end of the paper I present an alternative formulation where all UD have the same IT capabilities and they are heterogeneous in the utility they derive from the code.

motivated to participate, just by assuming that l could be a negative number. In order to have a well defined $G^{-1}(.)$ we will assume that G is continuous and strictly increasing.

At each stage j the decision of a UD to use C_j will depend on his net utility $S - \alpha_j l$ compared to an outside option. The outside option will be 0 if no alternative software exists but will be positive with competition.

When S is obtained the firm gets some income. The amount of income will depend on the way the code was developed. Under Proprietary Source (PS) development the firm gets V, while under Open Source (OS) development gets δV with $0 < \delta < 1$. The parameter δ represents a shortcut to the idea that the firm E, although it is not able to sell S, can still get some revenue from selling support services to EU or by selling some proprietary software complementary to S.

On the cost side, if the firm E develops the code as PS, it has to pay for each individual working on the code development, while under OS development the firm gets free help from m_j individuals of the UD community. Therefore the OS probability of advancing to the next step is:

$$\Phi(N_j + m_j) = 1 - (1 - q)^{N_j + m_j}$$

where m_j is the amount UDs devoted to the project at stage j while N_j is the amount of programmers paid by the firm. Of course under PS development $m_j = 0$. Once decided between OS and PS development, the firm must decide at each stage j the amount N_j of programmers hired at wage W.

Before we move to the model we are going to apply a useful transformation that will give us nicer expressions. We define $n_j = \beta N_j$ where $\beta = -\ln(1 - q) \in (0, 1)$, so the firm instead of choosing N_j it chooses n_j and pays wage $w = \frac{W}{\beta}$. With this transformation if the firm hires N_j programmers paying them WN_j and receives the free help from m_j UDs, it would be equivalent to say it hires n_j paying them wn_j and receives the free help from βm_j UDs. The probability of advancing to the next step given by

$$\Phi(N_j + m_j) = 1 - (1-q)^{N_j+m_j} = \phi(n_j + \beta m_j) = 1 - e^{-(n_j+\beta m_j)}$$

In the rest of the paper we will just say the firm hires the amount n_j of programmers. Finally we will define

$$\gamma_j = e^{-\beta m_j} < 1$$

so that

$$\phi(n_j + \beta m_j) = 1 - e^{-n_j}\gamma_j$$

3 The simplest setting: Two stages and no competition

In this case the code C_0 has to advance to C_1 and S is obtained. Let us first solve the problem of a OS firm. In order to decide the amount n_0 of programmers to hire it must compute first the amount of help the firm will receive from the UDs if the firm develops as OS. The utility per period derived by a UD at each stage (0 and 1) is given by:

$$U_0 \begin{cases} S - \alpha l & \text{if use } C_0 \\ 0 & \text{if not} \end{cases}$$

$$U_1 = S$$

The expressions for U_0 and U_1 follow easily from the basic framework description. Since we assumed that the finished software S does not require any programming time, the associated desutility l is 0. This also imply that all UDs will adopt the OS software at stage 1.

A UD will participate in OS development at stage 0 if

$$\begin{aligned} 0 &< S - \alpha l => \\ l &< \frac{S}{\alpha} \end{aligned}$$

The amount of UDs participating in OS development will be

$$m_0 = G(\frac{S}{\alpha})M$$

In particular if we assume that $G \sim U(0, \bar{l})$ then

$$m_0 = \frac{S/\alpha}{\bar{l}}M$$

As expected, the amount of development help from UD will be higher, if the net utility UD derive in OS is high ($S \uparrow$ or $\alpha \downarrow$) and if M is high.

Under OS, the profit maximization problem for firm E is:

$$\max_{n_0} \Pi_0 = (1 - e^{-n_0}\gamma_0)\delta V - wn_0$$

The first order conditions yields

$$\begin{aligned} 0 &= \gamma_0 \delta V e^{-n_0} - w \\ e^{n_0} &= \frac{\gamma_0 \delta V}{w} \end{aligned}$$

To make the problem interesting, we avoid having $n_0 \leq 0$ by making the following assumption

Assumption 1 $\gamma_0 \delta V > w$

$$n_0^{os} = \ln\left(\frac{\gamma_0 \delta V}{w}\right) = -\beta m_0 + \ln\left(\frac{\delta V}{w}\right)$$

The second term of n_0^{os} is just the amount of labor the firm would hire if the final income is δV and it received no help from the community. Given that the amount m_0 of UDs participating in OS the firm perfectly offsets this help. Assumption 1 could be rearranged as

$$m_0 < \frac{1}{\beta} \ln \left(\frac{\delta V}{w} \right)$$

In words, we are ruling out that the help provided by the community, although it might be significant, can never be such that the firm has a passive role while the whole developing effort comes from the community.[8]

The probability that the code advances will be given by

$$\begin{aligned} \phi(n_0^{os}) &= \left(1 - \gamma_0 e^{-\ln\left(\frac{\gamma_0 \delta V}{w}\right)}\right) \\ &= \left(1 - \frac{w}{\delta V}\right) \end{aligned}$$

so the expected profit under OS development will be

$$\begin{aligned} \Pi_0^{os} &= \left(1 - \frac{w}{\delta V}\right) \delta V - w \ln \left(\frac{\gamma_0 \delta V}{w} \right) \\ &= \delta V - w \left(1 + \ln \left(\frac{\gamma_0 \delta V}{w} \right)\right) \\ &= \delta V - w \left(1 + n_0^{os}\right) \end{aligned}$$

The profit maximization problem under PS is:

$$\max_{n_0} \Pi_0 = (1 - e^{-n_0})V - w n_0$$

which yields the following results

$$\begin{aligned} n_0^{ps} &= \ln \left(\frac{V}{w} \right) \\ \phi(n_0^{ps}) &= \left(1 - \frac{w}{V}\right) \\ \Pi_0^{ps} &= V - w \left(1 + n_0^{ps}\right) \end{aligned}$$

At this point, some observations should be pointed out.

[8] This assumption seems reasonable since the "sponsor" firm usually has an active role in spinout projects.

Proposition 1 *The firm E hires a lower amount of programmers n under OS*

Comparing n_0^{os} with n_0^{ps} we can see where does this reduction comes from. First because the revenue the firm gets at the final stage is now lower (δV_A instead of V_A). Second, there is a substitution effect: the firms perfectly offsets the help m_0 from the community of UD.

Proposition 2 *Under the assumption 1, the overall probability of obtaining the software S is smaller under OS.*

Since firm E perfectly offsets the amount of help provided by the UDs, the smaller probability comes exclusively from the lower revenue at the final stage (δ) that reduces the labor provision of the firm.

An interesting policy observation steams from this proposition: Although EU might face lower software expenditures with OS, the probability that the software is obtained is lower. Therefore EU might end up worse under OS development compared with PS development. If the weight of EU is sufficiently high, they might offset any welfare gain from the firm and the UDs, so PS development might be socially desirable.

OS will outperform PS if

$$w\left(n_0^{ps} - n_0^{os}\right) > (1-\delta)V$$

$$\begin{aligned} \ln\left(\frac{V}{w}\right) - \ln\left(\frac{\gamma_0 \delta V}{w}\right) &> \frac{(1-\delta)V}{w} \\ \ln\left(\frac{V}{w}\right) - \ln\left(\frac{e^{-\beta m_0}\delta V}{w}\right) &> \frac{(1-\delta)V}{w} \\ m_0 &= G(\frac{S}{\alpha})M > \frac{1}{\beta}\left(\frac{(1-\delta)V}{w} + \ln\delta\right) = \widetilde{m} \end{aligned}$$

Notice that the threshold value $\widetilde{m}$ depends positively on the income difference in terms of wage between OS and PS : $\frac{\lambda V - \delta\lambda V}{w}$. If this difference is small ($\delta \to 1$, $w \uparrow$ or $V \downarrow$) the firm will be willing to switch to OS development even for low values of community help.

In the same fashion, $\widetilde{m}$ depends negatively β or in other words is negatively related to the individual probability of success q: If each individual is more successful, the firm will be willing to switch to OS even for low values of help. Recall also that if the net utility UDs derive in OS is high ($S \uparrow$ or $\alpha \downarrow$) and if M is high, m_0 will be large and this increases the chances that OS outperforms PS development.

Proposition 3 *OS development will outperform PS development if the amount of individuals provided by the community m_0 exceeds the threshold value $\widetilde{m}$. Signs $\frac{\partial m_0}{\partial \alpha} < 0$, $\frac{\partial m_0}{\partial S} > 0$, $\frac{\partial m_0}{\partial M} > 0$ and $\frac{\partial \widetilde{m}}{\partial \delta} < 0, \frac{\partial \widetilde{m}}{\partial V} > 0, \frac{\partial \widetilde{m}}{\partial w} < 0, \frac{\partial \widetilde{m}}{\partial \beta} < 0$ are obtained under assumption 1.*

4 Adding stages

When there are only two stages, the development (one stage) could only be made under OS or PS. However if we add stages the firm might be willing to switch from one to the other. In general OS licences in order to foster collaboration and adoption they usually forbid the code owner to make the code proprietary in the future. Therefore we are reasonably assuming that this is not a possibility in our model. However we can allow the firm to start the project as PS and then switch to OS development. A natural question is wether this is optimal for the firm or not

Proposition 4 *The firm will never find profitable to start as PS and then turn to the OS development*

This is a natural result from our modeling setup. A project that starts as OS versus one that starts as PS and then changes to OS will have the same final income but the later will have higher developing costs. A simple way to illustrate this is to analyze the three stage case (C_0 must advance to C_1 and then to C_2 where S is obtained). The two developing stages could be done as OS or alternatively the first one as PS and the second one as OS. For both cases the problem at the final stage is the same:

$$\max_{n_1} \Pi_1 = (1 - e^{-n_1}\gamma_1)\delta V - wn_1$$

The optimal n for both cases is given by $n_1^{os} = -\beta m_1 + \ln\left(\frac{\delta V}{w}\right)$. The expected profit is: $\Pi_1^{os} = \delta V - w\left(1 + n_1^{os}\right)$.

In the first stage the problem differs. If the firm develops OS the problem is:

$$\max_{n_0} \Pi_0 = (1 - e^{-n_0}\gamma_0)\Pi_1^{os} - wn_0$$

The optimal n is $n_0^{os} = -\beta m_0 + \ln\left(\frac{\Pi_1^{os}}{w}\right)$. Recall that $m_j = G(\frac{S}{\alpha_j})M$. Since $\alpha_0 > \alpha_1$ and $G(.)$ is an increasing function then $m_0 < m_1$. So

$$\Pi_0^{os} = \Pi_1^{os} - w(1 + n_0^{os}).$$

If the firm develops the first step as PS then the problem is

$$\max_{n_0} \Pi_0 = (1 - e^{-n_0})\Pi_1^{os} - wn_0$$

The optimal n is $n_0^{ps} = \ln\left(\frac{\Pi_1^{os}}{w}\right)$.So:

$$\Pi_0^{ps} = \Pi_1^{os} - w(1 + n_0^{ps}).$$

Since $n_0^{ps} > n_0^{os}$ then $\Pi_0^{ps} < \Pi_0^{os}$.

Because OS development produces less final income, the whole point in choosing OS is to save on developing cost. Therefore the firm always values the help m provided by the OS community. A key element in this analysis is that this m is decided at each period and does not depend on the history or the future of the project since the UD's opportunity cost is 0.

Another thing to notice is that for PS or OS development it is always the case that $\Pi_j < \Pi_{j+1}$. Then if we compute at any $j-1$ the difference between PS and OS we have that

$$\Pi_{j-1}^{ps} - \Pi_{j-1}^{os} = \Pi_j^{ps} - \Pi_j^{os} + w(n_{j-1}^{os} - n_{j-1}^{ps})$$

so

$$\Pi_{j-1}^{ps} - \Pi_{j-1}^{os} = \Pi_j^{ps} - \Pi_j^{os} + w\left(\ln\left(\gamma_{j-1}\frac{\Pi_j^{os}}{\Pi_j^{ps}}\right)\right)$$

From this expression we can see that, since $\gamma_{j-1} = e^{-\beta m_{j-1}} < 1$, the difference between $\Pi_{j-1}^{ps} - \Pi_{j-1}^{os}$ if positive, is reduced as we move to stage 0. Therefore

Proposition 5 *Given two projects with the same final V, if one of them has more stages than the other it is more likely that OS development will outperform PS development, if the pool of UD is sufficiently high.*

Therefore, controlling for V, we would tend to see that projects requiring long development periods are carried out as OS.

5 Adding competition. The entry game.

The framework described above assumes that the code development is made in isolation, however, in many cases the OS software development occurs under the competition of an existing proprietary alternative.

To understand how a proprietary alternative can affect the development of an OS alternative we will consider a two period entry game. We assume that a Firm I is already selling a developed proprietary software in the market. Firm E must decide to enter or not the market and whether to develop a software as OS or PS. To make the problem interesting we assume that Firm I can price discriminate between sophisticated users (potential UD of a OS alternative) and unsophisticated ones (EU). We also consider that the Firm E faces some entry costs F. To keep the problem simple we continue to assume a two stage developing process (i.e. $C_0 \to C_1$ and S is obtained)

5.1 Timing

The timing of the problem is as follows

t=0

1. **Incumbent** PS firm I, decides price p charged to UDs. p is observed by Firm E and UDs

2. **Firm** E decides to pay cost F to enter the market or not. If enter, E choose between OS or PS development and hires the amount of programmers n accordingly.

3. **UD's** buy firms I software or use firm E code (if E enters and develops OS) depending on their programming desutility l. Depending on their decision, first period utility $U_{t=0}$ is realized.

t=1

Payoffs for firms I and E and UD's second period utility $U_{t=1}$ are realized. Payoffs depend on the fact that firm's E code has advanced or not.

5.2 Payoffs

The payoffs of the different agents are as follows:

a If firm E enters as PS and succeeds, the final income V is shared in the following way, $\lambda^{ps}V$ goes to firm E and $(1-\lambda^{ps})V$ to firm I.

b If firm E enters as OS and succeeds, we will assume that the overall income of the industry is reduced in a proportion $(1-\delta)$ with $(\delta < 1)$ then $\lambda^{os}\delta V$ goes to firm E and $(1-\lambda^{os})\delta V$ to firm I. The variable δ captures the idea that the competition in the EU market is tougher under OS. Since OS development entails lower costs we will assume, to make the problem interesting, that $\lambda^{ps}V > \lambda^{os}\delta V$. Moreover, to simplify expressions we will assume that $\lambda^{os} = \lambda^{ps} = \lambda$. The parameter λ is just a shortcut to the result of a market game in $t = 1$.

c The utility of a UD derived at each period from buying the PS software of firm I is:

$$U^I_{t=0}\begin{cases} S-p & \text{if buy} \\ 0 & \text{if not} \end{cases}$$

$$U^I_{t=1}\begin{cases} S & \text{if bought in 0} \\ 0 & \text{if not} \end{cases}$$

d The utility of UD derived from using OS of firm E is

$$U^{os}_{t=0}\begin{cases} S-\alpha l & \text{if use } C_0 \\ 0 & \text{if not} \end{cases}$$

$$U^{os}_{t=1}\begin{cases} S & \text{if code advanced} \\ S-\alpha l & \text{if code not advanced} \\ 0 & \text{otherwise} \end{cases}$$

Notice that $U^{os}_{t=1}$ is the utility at period $t=1$ and not the utility at stage 1 as we considered before. Therefore at period $t=1$ if the OS project has not been successful the UD will derive stage 0 utility. Another important thing is that Firm E entry threat already makes UDs better off: if firm I faced no entry it could extract all the UD surplus by charging the price $p^m = 2S$, however the maximum price for which firm I has some demand is lower that p^m since UD can always wait until $t=1$ to get the software.

In particular, under OS development, if a UD with $S < \alpha l$ waits until $t=1$ he has an expected surplus of $\phi^{os}S$ so the maximum price I can charge at $t=0$ is $p^{\max os} = 2S - \phi^{os}S$. If I sets $p > p^{\max os}$ it would face no UD demand. Since the expected UD surplus is positive when buying I software we conclude that if E develops as OS, the whole population M of UDs will be divided between those that adopt OS and those that buy I's software at $t=0$.

We are not defining a $U^{ps}_{t=1}$: if firm E develops as PS and it is successful the software would appear in the market at $t=1$ and firm I can always fix a price such that all UDs buy at $t=0$.Under reasonable conditions, the price that deters OS development is sufficiently low such that all UDs buy at $t=0$.

Having defined the payoffs at $t=1$, to solve the game we first solve the decision problem of the UDs and firm E optimal level of n. Finally determine the optimal level p of firm I.

5.3 Determination of m

If the firm E has entered as OS, a UD will choose it if:

$$S - \alpha l + \phi^{os}(S) + (1 - \phi^{os})(S - \alpha l) > 2S - p$$

$$l < \frac{p}{(2 - \phi^{os})\alpha} = \widehat{l}$$

The UD with $l = \widehat{l}$ has desutility high enough such that he is indifferent between paying or spending time on OS. The whole population of UD is divided between those with low programming desutility $l < \widehat{l}$ that adopt the OS, and those with high desutility $l > \widehat{l}$ that buy software from firm I.

This implies the following amount of labor available to firm E:

$$m(\phi^{os}, p) = G\left(\frac{p}{(2 - \phi^{os})\alpha}\right) M$$

Since we have assumed that $\frac{\partial G}{\partial l} > 0$ The expected signs $\frac{\partial m}{\partial p} > 0, \frac{\partial m}{\partial \phi} > 0, \frac{\partial m}{\partial \alpha} < 0$ are obtained.

Proposition 6 *The amount of UD help the OS project receives is increasing in the price of the proprietary alternative, increasing in the in the probability ϕ^{os} of advancing to the next step and increasing in the net utility the code provides at the developing stage.*

5.4 Determination of n, ϕ and $\widetilde{p}$

For a given amount m we can compute the optimal choice of labor n_0^{os} by the OS firm. To avoid negative n_0^{os} we need again

Assumption 2 $\gamma\delta\lambda V > w$

$$\begin{aligned} n_0^{os}(m) &= \ln\left(\frac{\gamma\delta\lambda V}{w}\right) \\ &= \ln\left(\frac{e^{-\beta m}\delta\lambda V}{w}\right) \\ &= -\beta m(\phi,p) + \ln\left(\frac{\delta\lambda V}{w}\right) \end{aligned}$$

for a given m and n the probability that the code advances next step is

$$\phi^{os} = \left(1 - \frac{w}{\delta\lambda V}\right)$$

Then the probability ϕ^{os} does not depend on n or m itself. Therefore for a given p we solve for m and n_0^{os}. The profit of firm E developing as OS is

$$\begin{aligned} \Pi_0^{os}(p) &= \left(1 - \frac{w}{\delta\lambda V}\right)\delta\lambda V - w\ln\left(\frac{\gamma(p)\delta\lambda V}{w}\right) - F \\ & \delta\lambda V - w\left(1 + n_0^{os}(p)\right) - F \end{aligned}$$

If the firm decides to develop as PS the optimal amount of labor is given by

$$n_0^{ps} = \ln\left(\frac{\lambda V}{w}\right)$$

so

$$\phi^{ps} = \left(1 - \frac{w}{\lambda V}\right)$$

Of course, since the firm is no longer relying on the community of UD's, the amount of labor it hires does not depend on the price p the firm I charges to UD's. Therefore PS profits will not depend on p :

$$\begin{aligned}\Pi_0^{ps} &= \left(1-\frac{w}{\lambda V}\right)\lambda V - w\ln\left(\frac{\lambda V}{w}\right) - F \\ &\lambda V - w\left(1+\ln\left(\frac{\lambda V}{w}\right)\right) - F \\ &\lambda V - w\left(1+n_0^{ps}\right) - F\end{aligned}$$

Notice also that $\phi^{ps} > \phi^{os}$, so competition has not affected proposition 2.

Comparing Π_0^{ps} and $\Pi_0^{os}(p)$, firm E will prefer OS if

$$m_0(p) > \frac{1}{\beta}\left(\frac{(1-\delta)\lambda V}{w} + \ln\delta\right) = \widetilde{m}$$

Except for λ this is the same expression we found in section 3. Recall that $\frac{\partial\widetilde{m}}{\partial\delta} < 0$, $\frac{\partial\widetilde{m}}{\partial V} > 0$, $\frac{\partial\widetilde{m}}{\partial w} < 0$, $\frac{\partial\widetilde{m}}{\partial\beta} < 0$. We also have $\frac{\partial\widetilde{m}}{\partial\lambda} > 0$.

The p that makes firm E indifferent between OS and PS is

$$\begin{aligned}m_0(p) &= \frac{1}{\beta}\left(\frac{(1-\delta)\lambda V}{w} + \ln\delta\right) = \widetilde{m} \\ G\left(\frac{p}{(2-\phi^{os})\alpha}\right)M &= \widetilde{m} \\ \widetilde{p} &= G^{-1}\left(\frac{\widetilde{m}}{M}\right)(2-\phi^{os})\alpha\end{aligned}$$

From the expression above we can see that $\widetilde{p}$ is positive. The expression is quite intuitive: notice that the level $\widetilde{p}$ to prevent OS will be small if the threshold level $\widetilde{m}$ is small compared to the size of the UD market or, as stated before, if the income difference in terms of wage between OS and PS is small. Also $\widetilde{p}$ will be small if the probability of obtaining the software is higher or if net utility the code provides at the developing stage is smaller.

Proposition 7 *If the firm I, charges a price $p \leq \widetilde{p}$ and the firm E decides to enter, the code will be developed as PS. If the firm I charges a price $p > \widetilde{p}$ and the firm E decides to enter, the code will be developed as OS. Signs $\frac{\partial\widetilde{p}}{\partial\delta} > 0$, $\frac{\partial\widetilde{p}}{\partial\alpha} > 0$ and $\frac{\partial\widetilde{p}}{\partial\phi} < 0$ are obtained.*

To make the problem interesting we are going to assume that OS is a viable development strategy. That is to say $m_0(p^{\max os}) = \frac{S}{\alpha} > \widetilde{m}$.

5.5 Determination of p

The optimal problem of the incumbent firm I is to decide p.The minimum price firm I would charge is $\widetilde{p}$: under PS development setting p further down has no effect on firm E's profit since it is not relying on UDs to develop the code. A price $p < \widetilde{p}$ would only reduce firm I's profits on UDs. Therefore:

Proposition 8 *entry deterrence using p as an instrument is not possible if at price $\widetilde{p}$, firm E profits $\Pi_0^{ps}(\widetilde{p}) = \Pi_0^{ps}(\widetilde{p})$ are positive. If F is such that $\Pi_0^{ps}(\widetilde{p}) = \Pi_0^{ps}(\widetilde{p})$ are negative then $p^d \geq \widetilde{p}$ exists such that OS development is deterred.*

If we define $\widetilde{F}$ as the level of F such that $\Pi_0^{ps} = \Pi_0^{os}(\widetilde{p}) = 0$

$$\widetilde{F} = \lambda V - w\left(1 + n_0^{ps}\right)$$

Then the proposition above is telling us that if $F \geq \widetilde{F}$ firm I is facing the choice between inducing OS development or deterring entry. If $F < \widetilde{F}$ then firm I is facing the choice between inducing PS or OS development. Then the profits of I are given by:

If $F \geq \widetilde{F}$

$$\Pi^I \begin{cases} \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V + p(1-G(p))M & \text{for } p > \widetilde{p} \text{ such that } E \text{ enter and chooses OS} \\ V + pM & \text{for } p \geq \widetilde{p} \text{ such that } E \text{ does not enter} \end{cases}$$

If $F < \widetilde{F}$

$$\Pi^I \begin{cases} \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V + p(1-G(p))M & \text{for } p > \widetilde{p} \text{ such that } E \text{ chooses OS} \\ \phi^{ps}(1-\lambda)V + (1-\phi^{ps})V + pM & \text{for } p = \widetilde{p} \text{ such that } E \text{ chooses PS} \end{cases}$$

5.5.1 Case $F \geq \widetilde{F}$

If $F \geq \widetilde{F}$ entry deterrence is possible, it will be optimal for firm I to induce it if:

$$V + p^d M > \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V + p^{nd}\left(1 - G(p^{nd})\right)M$$

Where p^d denotes deterrence price and p^{nd} no deterrence price. Notice first that because ϕ^{os} does not depend on p, the p that maximizes the RHS comes from

$$\max_p p\left(1 - G(p)\right)M$$

so the F.O.C gives us

$$\left(1 - G(p^{nd})\right) - p^{nd}\frac{\partial G(p^{nd})}{\partial p} = 0$$

or in terms of price elasticity

$$\left(1 - G(p^{nd})\right)(1 + \epsilon_p) = 0$$

Since $(1 - G(p))$ is positive, then the optimal p^{nd} is such that makes $\epsilon_p = -1$

On the other side p^d is the highest price that makes $\Pi_0^{os}(p^d) = 0$. If we define $\widehat{m}$ as the level of m that yields 0 profit for firm E :

$$\widehat{m} = \frac{1}{\beta}\left(\ln\left(\frac{\delta\lambda V}{w}\right) - \frac{\phi^{os}\delta\lambda V - F}{w}\right)$$

then p^d is just:

$$p^d = G^{-1}\left(\frac{\widehat{m}}{M}\right)(2 - \phi^{os})\alpha$$

Since $V > \phi^{os}(1 - \lambda)\delta V + (1 - \phi^{os})V$, it is required for deterrence to be optimal that $p^d > p^{nd}(1 - G(p^{nd}))$ and this is not always true.

If $p^d > p^{nd}$ the condition for deterrence will hold trivially. This could happen if for example firm I 's UDs demand is very sensitive to price changes ($\frac{\partial(1-G(p))}{\partial p}$) or if $\widehat{m}$ is very high.[9]

It might be more interesting instead, to think on what is needed to make entry deterrence **not** optimal:

[9]If we assume for example that $G \sim U(0, \bar{l})$, then we would have: $p^d = \bar{l}\left(\frac{\widehat{m}}{M}\right)(2 - \phi^{os})\alpha$ and $p^{nd} = \frac{\bar{l}}{2}(2 - \phi^{os})\alpha$ so to have $p^d > p^{nd}$ we need $\frac{\widehat{m}}{M} > \frac{1}{2}$. The condition $p^d > p^{nd}(1 - G(p^{nd}))$ is verified if $\frac{\widehat{m}}{M} > \frac{1}{4}$,

$$V + p^d M < \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V + p^{nd}\left(1 - G(p^{nd})\right) M$$

Again, since $V > \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V$, then $p^d M < p^{nd}\left(1 - G(p^{nd})\right) M$ must be big enough such that the inequality is reversed. Then, not only $p^{nd}\left(1 - G(p^{nd})\right) - p^d > 0$ is needed but also M should be big enough. This suggests that it might be more likely to observe entry accommodation of a OS alternative when the market of sophisticated UD is rather big (compared to EU) and on the contrary a more aggressive behavior should be expected if this market is rather small.

Proposition 9 *If* $p^{nd}\left(1 - G(p^{nd})\right) > p^d$, *entry accommodation of an OS alternative is more likely, the larger the share of sophisticated users in the market.*

5.5.2 Case $F < \widetilde{F}$

Finally if entry deterrence is not possible, then firm I must decide whether to set $p = \widetilde{p}$ to trigger PS development or set $p > \widetilde{p}$ and have OS development.

PS development will be preferred if

$$\phi^{ps}(1-\lambda)V + (1-\phi^{ps})V + \widetilde{p}M > \phi^{os}(1-\lambda)\delta V + (1-\phi^{os})V + p^{nd}(1 - G(p^{nd}))M$$

or

$$(\phi^{ps} - \phi^{os}\delta)(1-\lambda)V - (\phi^{ps} - \phi^{os})V + (\widetilde{p} - p^{nd}(1 - G(p^{nd})))M > 0$$

Several trade offs work at the same time and might offset each other. First notice the second term $-(\phi^{ps} - \phi^{os})V$: since $\phi^{ps} > \phi^{os}$, firm I faces a lower probability of competition in the future under OS development, and this works against inducing PS development. On the other hand, the first term $(\phi^{ps} - \phi^{os}\delta)(1-\lambda)V$ captures the fact that if OS project succeeds it reduces industry income due to parameter δ, so although less probable than PS, OS hurts more I profits. Therefore the difference between PS and OS expected income on EU remains

undetermined. The third term captures the profits on the UDs side: if $\widetilde{p} > p^{nd}(1 - G(p^{nd})$ this term will work in favor of PS development. However we face a similar issue as in the $F \geq \widetilde{F}$ case, given that p^{nd} is set to make $\epsilon_p = -1$ so $p^{nd} \lesseqgtr \widetilde{p}$.

The main intuition from this section is that, despite the fact that low prices to sophisticated users (compared to final users) by incumbent firms might be natural due to price elasticity, it might also hide some entry/OS development deterrence. The key point is that the incumbent firm by lowering the price to UDs is taking away development help provided by the community to the OS alternative. Rather that reducing the entrant future income, the incumbent is deterring by increasing the entrant development cost.

6 Alternative formulation

UD instead of being heterogenous in l they could differ in the utility they derive from the code. The utility derived from a finished software S is θ. If the software is developed as OS, the **gross** utility derived from using the code at stage j is $\alpha_j\theta$ with $0 < \alpha_j < 1$. If UDs have to spend some programming time, normalized to 1, to adapt the code. Therefore the **net** utility of using the code at stage j is

$$\alpha_j\theta - 1$$

The total amount of UD is given by M, and the fraction that has θ higher than $\widetilde{\theta}$ is $1 - F(\widetilde{\theta})$. The decision of a UD to participate in OS will depend on $\alpha_j\theta$ compared to 1 and to its outside option. At each stage only individuals such that $\theta > \frac{1}{\alpha_j}$ will participate.

A variation on this could be that the net utility is $\alpha_j\theta t - \frac{1}{2}t^2$. And t is the amount of time the UD spends in the project. The optimal amount of time at each period would be given by $t^* = \alpha_j\theta$. Individuals with high θ are those who participate more in the project, and this participation is increasing in at each stage since $\alpha_{j+1}\theta > \alpha_j\theta$

6.1 Extension: Dynamics of cooperation

The alternative formulation presented before results very useful to understand the dynamics of UDs collaboration. As we saw in the introduction of the basic framework, UDs have to spend time to be able to use the code since it is not "finished": it might contain "bugs" or it might need the development of new features to be useful. Along the paper we assumed that UDs are willing to share their code modifications and face no cost in doing so. Here we propose a new perspective that adds up to the literature about motives to participate in OS projects. We develop the idea that a UD will share his code modifications because if their contributions are included in the next version of the code, then they will need to spend less time to be able to use it.

The key ingredients that trigger collaboration are the sequential code improvement (which makes the newer versions of the code more desirable) and the savings in programming time to be able to use the code.

This is modeled assuming that in the next period it will be necessary to spend only ρ $(0 < \rho < 1)$ units of time to be able to use the code (ρ can be interpreted as a probability or the fraction of what was included in the code). Contributing, however, will not be free: it implies a desutility c (this could be interpreted for example as time spent submitting bug findings or code modifications). To simplify the analysis we are going to assume that the savings last one period and they are not permanent.

The utility of a UD at stage j and $j+1$ is given by

$$U_j^{os} \begin{cases} \alpha_j\theta - 1 - c & \text{if use OS and contribute} \\ \alpha_j\theta - 1 & \text{if use OS and not contribute} \\ 0 & \text{otherwise} \end{cases}$$

$$U_{j+1}^{os} \begin{cases} \alpha_{j+1}\theta - \rho & \text{if contributed at } j \\ \alpha_{j+1}\theta - 1 & \text{if not contributed at } j \\ 0 & \text{otherwise} \end{cases}$$

UDs will collaborate if savings are big enough

$$c \leqslant \phi(1-\rho)$$

The UD with the minimum θ that will collaborate is

$$\widetilde{\theta} = \frac{c + \phi\rho + 1}{\alpha_j + \phi\alpha_{j+1}}$$

assuming the worst situation, that is a c such that $c = \phi(1-\rho)$ then

$$\widetilde{\theta_w} = \frac{1}{\alpha_j} \frac{\phi + 1}{\phi + \frac{\alpha_{j+1}}{\alpha_j}}$$

the minimum θ that would use the code in a world without collaboration is just

$$\widehat{\theta} > \frac{1}{\alpha_j}$$

since α_j and $\alpha_{j+1} \in (0,1)$ and $\alpha_j < \alpha_{j+1}$ then

$$\widetilde{\theta_w} < \widehat{\theta}$$

Future cost reductions, generates that some UD with $U_j^{os} < 0$ engage in OS

Proposition 10 *The possibility of cost reductions in the use of future stages of the code, could imply that some UDs with current negative utility from the use of the code, participate in OS development.*

This proposition might help to explain why some UDs or firms are attracted to collaborate with OS projects even at very early stages where the code seems of very little use for them: If the code progresses they expect to use it. If the future version of the code incorporates the modifications they have costly shared with the community in the past, then the code will be less costly for them to use when the moment comes (i.e. it will require less modifications and maintenance or it will have less compatibility problems, etc.).

I believe that this explanation about motives to participate fits well with MySQL and JBoss stories. In this cases, besides bugs reports, UDs code contributions rather than being

focused towards the core of the program, are aimed towards extensions they manly use for their own purposes (but that can be of use for others).

7 Related Literature

The paper that is more closely related to our way of modeling OS v.s. PS code development is Aghion, Dewatripont and Stein (2005). Their aim is to clarify the respective advantages and disadvantages of academic and private-sector research. As in our model an idea must overcome a fix number of stages to become a marketable product. With their model they determine the optimal path of academia/private sector stages the idea has to follow. While academia's (low focus) creative freedom implies a lower probability to advance stages, private research higher focus is more costly. Academia research is less expensive since lower wages paid to academic scientists reflect their willingness to forgo earnings in exchange for academic freedom. Their main finding is that private sector's "expensive" focus strategy only pays in later-stage research.

An important source of stylized facts on OS firms comes from Dahlander (2005). This paper provides a multiple case study on OS firms and the main goal is to address how OS firms generate returns and how that changes over time. From the six cases of small firms in Sweden and Finland, two of them where particularly useful to our model. One of the cases is related to a well established "second generation" OS firm that produces a database software (MySQL). The second firm exploits a webserver software solution (Roxen). Both OS projects were started by the firms and the software was developed using the support of UD. Firms now make profits on a combination of support services, software installation and customization (making the software "fit the customers") and from selling licences on complementary software. The paper stress the importance of being first movers and building a significantly big "community" that will help on the product development. This is closely related to our threshold value $\widetilde{m}$ that makes OS development feasible.

The idea that firms might collaborate with and benefit from OS is not new in the OS liter-

ature. Schmidtke (2006) views OS phenomena as the private provision of a public good. The model suggests that although improvements in such a non-excludable public good cannot be appropriated, companies can benefit indirectly from the OS "quality" in a complementary proprietary segment. The model assumes that OS software already exist (no code development is modeled) and the author is interested in crowding in/out form public investment and pricing strategies on the complementary good under market entry.

The idea of competition between OS and a PS alternative has also been studied. Sen (2005) analyses the competition game between a freely available open source software (OSS), the commercial version of the same (OSS-SS) and a proprietary software (PS). Two dimensions characterize the software: its network benefits and the usability. Conditions under which PS dominates the market are analyzed and OSS-SS is not always found to hurt PS alternative. Our model differs from this one because we rather study the pricing strategy of the PS firm towards the UD to prevent the development of the OS alternative. Verani (2006) builds a model were firms compete in a differentiated duopoly. Each firm sells a good made of two components, one of them a software. Demand depends on prices and quality of the "software" component. If goods are substitutes, the author finds that the investment in "software" quality is bigger under OS due to spillover effects across firms.

There has been a significant amount of literature to explain motivation of programmers to collaborate in OS development. Dewan et. al (2005) suggest a Principal (firm)-Agent (programmer) model with learning. The firm benefits from the programmer's participation in OS because he acquires skills that are useful for the firms own project. The firm cannot monitor the programmers effort division and too much OS attention might hurt firms project success. The programmer, on the other hand, wants to work in OS because it allows him to signal his talent to other firms and increase his wage. Spiegel (2005) also builds on a Principal - Agent model where programmers participate on OS to signal their talent to potential employers. In our model we point out another possible source of cooperation that differs from this literature: the dynamics of sequential code improvement and the possibility of saving programming time in future versions of the code.

Finally Athey and Ellison (2006) with a different modeling strategy also focus on the dynamics of OS. In their model, altruism and the anticipation of altruism are the key element that triggers OS development. The rate of decay of altruism jointly with the arrival of new programmers needs are the ingredients that govern the growth of the OS software. They also consider the effects of commercial competition on OS dynamics and they show commercial firms might reduce their prices to slow the growth of OS projects.

8 Conclusions

Although the model we presented here is a very stylized story of OS v.s. PS development we believe that it gives some interesting insights on the motivation of for-profit firms to start OS projects and when this is more likely to happen. High development cost, long development periods, low profits on EU and a significant population of UD are important ingredients to trigger OS development. We also find that the overall success probability is lower under OS development, this might hurt the welfare and should be taken in consideration when doing welfare analysis. The introduction of an existing PS alternative in the model allows to stress the importance of the UD community on the OS development. The incumbent firm by reducing prices charged to sophisticated users, is able to reduce the size of the community of UD that collaborate on the OS development and in this way it can deter entry or induce the PS development. Finally the dynamic structure of sequential code improvement provides an alternative explanation on user - developers collaboration.

Besides the existing results we consider that this set up has a significant potential for future extensions and improvements:

- Competition between UDs can be introduced to see to which extent the code development is affected and to study the rivalry conditions that allow OS success.

- The difference between UD and EU could be endogenized so the amount of EU could be made a choice variable for the firm (i.e.: a EU is just someone that verifies $\alpha S - l < 0$). If the firm's income comes mainly from selling support to EU, and the number of EU

is reduced as the OS progresses, the firm might choose not to develop the code too much.

- Issues on Governance structure could be studied under this setting (i.e. might the firm find profitable to forgo the control of the code development?)

9 References.

References

[1] Aghion, Philippe, Mathias Dewatripont, and Jeremy C. Stein, (2005), "Academic Freedom, Private-Sector Focus, and the Process of Innovation", NBER Working Paper No. 11542.

[2] Athey, S and G. Ellison (2006), "Dynamics of Open Source Movements". Working Paper.

[3] Bonaccorsi A., Rossi C., and S. Giannangeli (2004), "Adaptive entry strategies under dominant standards. Hybrid business models in the Open Source software industry'

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

[5] Dewan, Rajiv, Marshall Freimer, and Amit Mehra, (2005), "When a firm's employees work on open source projects"

[6] Lerner, Josh and Jean Tirole (2002),"Some simple economics of open source" Journal of Industrial Economics 50 no. 2, 197–234.

[7] Schmidtke, Richard, (2006), "Private Provision of a Complementary Public Good," Discussion Papers in Economics 964, University of Munich, Department of Economics.

[8] Sen, Ravi (2005). "A Strategic Analysis of Competition Between Open Source and Proprietary Software," Industrial Organization 0510004, EconWPA.

[9] Spiegel, Yossi (2005). "The Incentive To Participate In Open Source Projects: A Signaling Approach," Working Papers 05-23, NET Institute.

[10] West, J. and S. O'Mahony, (2005), "Contrasting Community Building in Sponsored and Community Founded Open Source Projects" Proceedings of the 38th Annual Hawai's International Conference on System Sciences, Waikoloa, Hawaii, January 3-6, 2005.

[11] Verani, Stephane, (2006). "Open Source Development in a Differentiated Duopoly," Economics Discussion / Working Papers 06-05, The University of Western Australia, Department of Economics.