UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Scienze Economiche "Marco Fanno"

DUAL LICENSING IN OPEN SOURCE SOFTWARE MARKETS

STEFANO COMINO
University of Udine

FABIO M. MANENTI
University of Padova

January 2010

"MARCO FANNO" WORKING PAPER N.112

Dual Licensing in Open Source Software Markets*

Stefano Comino† Fabio M. Manenti‡

January 2010

Abstract

In this paper we present a theoretical model to study the characteristics and the commercial sustainability of dual licensing, an open source (OS) business strategy that has gained popularity among software vendors. With dual licensing, a firm releases the same software product under both a traditional proprietary license and an open source one. We show that the decision to employ a dual licensing strategy occurs whenever the feedbacks of the open source community are valuable enough compared to the quality of the software that the firm is able to develop in-house. Our analysis points to the central role of an appropriate managing of OS licenses in order to balance the pros and cons of "going open source" and to make this versioning strategy viable for software vendors; our analysis also suggests a possible explanation for the observed proliferation of open source licenses.

J.E.L. codes: L11, L17, L86, D45.

Keywords: open source software, open source business models, embedded software, dual licensing, versioning, license proliferation.

*Much of this work has been conducted while the authors were visiting the School of Information of the University of California at Berkeley. We are extremely grateful to Hal Varian for his hospitality. Financial support from "Progetto di Ateneo" - Padova, 2005-07. Paper presented at the 2^{nd} FLOSS international workshop on Free/Libre open source software - Rennes 2008, at the 3^{rd} annual conference of the EPIP association - Bern, 2008, and at the 36^{th} annual conference of the EARIE - Ljubjana, 2009. The authors are grateful to Mark Schankerman, Laura Vici and Marcus Wagner for helpful suggestions and comments on earlier versions of the paper.

†Corresponding author: Dipartimento di Scienze Economiche, Università di Udine, Via Tomadini 30/A, 33100 UDINE (Italy), Tel. (39) 0432 249211, Fax. (39) 0432 249229, email: stefano.comino@uniud.it

‡Dipartimento di Scienze Economiche "M. Fanno", Università di Padova, Via del Santo 33, 35123 PADOVA (Italy), email: fabio.manenti@unipd.it

1 Introduction

Until recently, open source (OS) has been seen unfamiliar by the business community and, in many cases, it has been perceived as a real threat by commercial vendors. In the very last years, things have changed substantially and both large established incumbents such as IBM, HP or NEC as well as start-ups are increasingly embracing OS strategies.

Commercial firms may enjoy several benefits by "going open source". A firm may take advantage of the contributions of the community of OS developers either in the direct form of code enhancements or in terms of educated feedbacks and reviews received from expert users.[1] Furthermore, open source represents a powerful channel of software distribution: it may constitute a key strategic instrument to improve the perceived quality of the product and to enlarge the installed base of users, thus helping firms in establishing an industry standard.

The key issue for a software vendor is how to design a sustainable business model based on open source solutions, provided that various features of OS software development and distribution seem to be unappropriate for commercial exploitation.[2] In a recent study based on 218 companies that were collecting at least 25% of their revenues, directly or indirectly, from open source, Daffara (2009) observes that the most common OS business strategies fall into two main categories. The sale of services that are complementary to the open source software, such as customization, consulting, training and documentation, constitutes the first, and probably the most common, category of OS business strategies.[3] The sec-

[1] It deserves to be noticed that "non-code contributions" from the OS community are as important as "code contributions". For instance, Jullien (2006), in a study on the Open Cascade project, reports that: bug-fixing, preparation of documentation or tutorials and other contributions not directly linked with code writing represented the 20% of the value of the software. Similar findings, for other OS projects, are in Seigo (2006) and in Mueller (2007).

[2] For instance, OS licenses require the code of the software to be freely re-distributable; when releasing the software code, an individual, or the firm cannot prevent or restrict (e.g. by requiring royalties) its re-distribution (see article 1 of the Open Source definition `www.opensource.org/docs/definition.php`).

[3] Just to take a relevant example, in 2001, IBM started the open source project Eclipse in order to promote the use of the programming language Java within server products; IBM profited from selling related products such as components of WebSphere and WebLogic (West and Gallagher, 2004). Alternatively, software vendors often offer deployment support, customization and adds-on products for OS solutions; see Rajala et al. (2007) for a comprehensive discussion of the well-known Red Hat case.

ond most significant category is related to versioning strategies. In many cases, firms offer different versions of the software and profit from selling upgraded packages providing additional functionalities with respect to the open source version of the software. Within this second category, a business strategy which is peculiar to the software industry and that it is becoming increasingly popular among commercial vendors is *dual licensing*.[4]

With dual licensing firms mix traditional and OS-based strategies by offering the same software product under both a traditional proprietary license and an open source one; in the latter case, the software is typically provided for free or at a nominal fee. There are various reasons why customers, when offered a free OS version of a software, may still prefer to pay for the proprietary version; certainly, one of the most important reason accrues from the reciprocal provision imposed by some OS licenses: open source customers are required to redistribute their derived works under the same licensing scheme as the original software, including the requirement to make the source code of the derived software publicly available.[5] To better grasp this critical issue, it is useful to quote Oracle, the vendor of the embedded database BerkeleyDB; in its web page, Oracle describes its dual licensing strategy as follows:[6]

"Our open source license permits you to use Berkeley DB [...] at no charge under the condition that if you use the software in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. If you do not want to release the source code for your application, you may purchase a license from Oracle."

Commercial customers that use, modify and embed Berkeley DB into their own appli-

[4]Around 10% of the companies in Daffara's sample were employing dual licensing. Some notable examples of software packages released according to this commercial strategy are MySQL, Berkeley DB, Qt, and Asterisk. See Välimäki (2005) and Moody (2006) for a discussion of these and of other cases.

[5]Oracle suggests that beyond relieving from the reciprocal provision, there are additional benefits of adopting the proprietary version: in the description of its dual licensing strategy, Oracle argues that the proprietary version of the software includes "legal assurances, warranties, and a wide array technical and aftersale services provided by a full-time dedicated development team". Furthermore, many OS software projects are distributed under licenses that allow the licensor to terminate the agreement conditional on the occurrence of specific events, and this clearly puts the customer to a risk in case she/he needs to invest money and effort in using the software (see Rosen, 2004 for a discussion of the so-called "patent termination clauses").

[6]See `http://www.oracle.com/technology/software/products/berkeley-db/htdocs/licensing.html`.

cations might be reluctant to use the OS version. These applications may be products per se or, more frequently, they are part of a more complex system that customers produce and sell. In both instances, it is clear that since customers want to keep proprietary control on their derived products, they may be willing to pay in order to be relieved from the reciprocal provision imposed by the open source version.

The software vendor benefits from releasing for free the open source version thanks to the contributions of OS adopters. These contributions, either "code" or "non-code", are then incorporated into the proprietary version and this helps to ameliorate product's quality.[7] It deserves to be noticed that typically the software vendor keeps strong control on the open source project and maintains the possibility of re-using "code" contributions by requiring external programmers to grant the permission to incorporate the lines of code that they have written into the proprietary version.[8]

As argued in the paper, the licensing terms of the open source version of the software are pivotal in the commercial sustainability of a dual licensing strategy. On the one side, a restrictive license, e.g. a license that imposes the reciprocal provision, represents an important safeguard against the possible cannibalization of the proprietary version of the software since it discourages some potential customers from adopting the open source version.[9] On

[7]Sun Microsystems, the producer of MySQL describes its dual licensing strategy as follows: "We have over 4 thousand paying customers who have chosen the commercially-licensed MySQL server, and we have over 4 million users who use MySQL under the GNU General Public License (GPL). [...] Thanks to our commercial customers, we can afford to develop and improve the product at a fast pace. [...] And thanks to the huge user community, MySQL undergoes rigorous 'battle-testing'; see `http://mysql.com/news-and-events/newsletter/2003-11/a0000000220.html`.

[8]For instance Digium, the producer of the telecommunications software Asterisk, requires OS contributors to sign the "*Digium open source project submission agreement*"; according to this agreement, contributors "....grant Digium a perpetual, worldwide, royalty-free, irrevocable, non-exclusive, and transferable license to use, reproduce, prepare derivative works of, publicly display, publicly perform, distribute the Submissions, and to sublicense such rights to others. The rights granted may be exercised in any form or format, and Digium may distribute and sublicense to others on any licensing terms ..." (see `https://issues.asterisk.org/view_license_agreement.php`). In other cases, the vendor rewrites and reassembles the lines of code written by OS programmers and then includes them in the proprietary version of the software. As observed by Välimäki (2005), in MySQL AB project "All contributions are checked and rewritten by company developers..." (p. 212).

[9]Note that once the code has been released to the OS community, cannibalization may take the form of the so called *forking*: OS programmers might download the code and start independent development on it.

the other side, the licensing terms affect also the size of the OS community, as well as the incentives that OS programmers have in contributing to the software project. As documented in many empirical studies, the terms of distribution of an OS project are an important determinant of its overall progress; Comino, Manenti and Parisi (2007) have shown that OS projects released according to a more restrictive license are less likely to succeed. Others have shown that more restrictive licensing terms negatively affect the contribution (average lines of code written) of the members of the OS community (see and Fershtman and Gandal, 2007).

More specifically, in this paper we consider a profit maximizing firm that is developing a software project targeted to commercial customers. The firm either develops the project completely in-house or it employs a dual licensing strategy. In this latter case, it is crucial to manage appropriately the open source license in order to balance pros (the contributions of the OS community) and cons (the risk of cannibalizing the proprietary package) of "going OS". Assuming that customers have heterogenous preferences towards the restrictions imposed by the OS license, we derive the conditions under which dual licensing is profitable. Moreover, we discuss how an appropriate definition of the licensing terms allows the firm to optimally segment its potential customers into two groups: those who adopt the OS version, and that contribute to enhance the software quality, and those that pay for the proprietary version.

Our paper is related to various strands of economic literature. As this introduction should have made evident, dual licensing represents an example of versioning; many authors have shown that versioning may be a profitable strategy when it allows the firm to enlarge its market share and to sell also to customers with low quality evaluation (Shapiro and Varian, 1998; Belleflamme, 2005). Moreover, a bulk of papers has shown that, when the market is affected by consumption externalities, a firm may benefit from "creating" a competitor in order to expand the installed base of users. Economides (1996) and Gayer and Shy (2003) are two relevant examples of this literature; the former shows that a monopolist may profit from inviting entry of a compatible rival, while the latter found that a software developer may benefit from allowing piracy activity. In this paper, we show that versioning, with the annexed creation of a competitor, might be profitable also in the absence of any market

To prevent this risk, the firm must maintain a strong leadership in the management of the project. In the theoretical model, we do not account explicitly for the risk of forking.

enlargement effect and without the possibility to exploit the benefit of greater consumption externalities. In a context where customers are expert users who are also able to provide a significant contribution to the quality of the product, the benefits of releasing an open source version of the software accrue to development externalities. In this world characterized by OS adopters that become part of the production process, the better quality of the software achieved thanks to the efforts of the OS community goes to the advantage of all customers, those who purchase the proprietary version as well as those who adopt the open source one.

The theoretical literature on the "economics of open source" has been focussed mainly on modeling competition between open source and proprietary software; little has been done to achieve a better understanding of the rationales for commercial vendors to go open source (see Lanzi, 2009 for a recent review). An exception, closer to our paper, is represented by Mustonen (2005); the author models a firm's decision to support an existing open source community that is developing a rival program. The author shows that the firm may find it optimal to sustain OS when this promotes compatibility between the OS and the proprietary versions of the software, and when the OS community provides a sufficiently valuable development externality. Mustonens's development externality differs substantially from ours'.

In Mustonen (2005) the strength of the externality is exogenous, while in our setting it is endogenously determined and proportional to the size of the open source community. More importantly, in Mustonen (2005), the development externality is relevant only in relation to the existence of the OS alternative, which exists independently of what the firm does; on the contrary, if rather than supporting an existing project, it would be the firm that "creates" the OS project, as in our model, then Mustonen's development externality would not have any impact on firms' behavior.

The rest of the paper is organized as follows: in Section 2 we present the model and we derive the main results, while in Section 3 we conclude.

2 The model

Consider a commercial firm that has started developing a new project for an embedded software. The software is directed to commercial customers who then need to spend some time and effort in order to embed and tailor it to their own products; in doing so, they improve the original software by adding new functionalities or by simply fixing possible

bugs.

The firm faces an alternative in developing and distributing the software. It can either complete the project in-house, and then sell the software. Alternatively, it can endorse a dual licensing strategy in order to involve customers in the development of the project, thus improving the quality of the software.[10] In this case, the firm makes the open source version of the software available at no fee by posting it on a public repository, and benefits from the new functionalities and from the bug-fixing activity of those customers who adopt it (i.e. it benefits from the contributions of the open source community). At the same time, the firm profits from selling at a positive price a second version of the software which is distributed under proprietary licensing terms. In what follows, we will refer to these two versions of the software as the open source and the proprietary version respectively.

Formally, the firm takes sequentially the following decisions:

1. it decides whether to release the source code to the open source community (i.e. whether to employ a dual licensing strategy); if it releases the code, the firm sets the degree of restrictiveness of the OS licence: formally, it chooses $r \geq 0$;

2. once the code has been developed, the firm chooses the price p of the proprietary version.

In turn, customers observe the firm's licensing and pricing strategy and take their adoption decision.[11] They may adopt the software, either the proprietary or the OS version if

[10] As discussed in the Introduction, typically firms adopt versioning strategies to increase sales or to enlarge the installed base of users; in our setting, the benefit of dual licensing strategy (i.e. of releasing the open source version of the software) is not intended to enlarge the market in none of these ways, but at exploiting the so-called "development externality". In fact, we assume that customers have homogeneous preferences with respect to the quality of the software (they are heterogeneous only with respect to license restrictiveness); this implies that, in equilibrium, the market is always fully covered also when the firm does not release the open source version. The fact that dual licensing is not intended to increase sales is also supported by the practical observation that firms distribute for free the open source version.

[11] Note that according to the timing of the model some users may adopt the OS version at stage 1 and then contribute to the development of the code, while others will postpone their adoption decision after the proprietary version has been released. Assuming that customers *i*) do not derive additional benefits from adopting the OS version at stage 1 and *ii*) they rationally forecast the size of the OS community, then the exact timing of adoption decision is not relevant. Allowing customers to derive additional benefits from early

available, or they may choose not to adopt any software at all; in this last case they enjoy their reservation utility u_o.

Customers evaluate not only the quality of the software "per-se", but also the terms of licensing. Since they embed it into their own products, then, other things equal, they prefer to obtain the software under unrestrictive licensing terms. We assume that customers are heterogeneous with respect to license restrictiveness, and we parameterize their preferences with the term t; in particular, we assume that t is distributed according to the c.d.f. $F(t)$ over the support $[0, T)$, where $T > 0$ may be either finite or infinite.[12] Customers with a low t are little affected by license restrictiveness while customers characterized by a large t receive a strong disutility from r.

Formally, in case customer t chooses to adopt the OS version of the software, then she/he obtains a net benefit equal to:

$$U_{OS}(t, r) = V + \theta(r)N - tr,$$

where $V + \theta(r)N$ is the overall quality of the software, with V representing the quality developed by the firm and $\theta(r)N$ the development externality accruing from the OS community; $\theta(r)N$ is increasing in N, the mass of open source adopters, and in the strength of the externality, $\theta(r) \geq 0$. Finally, tr represents the disutility that the restrictions imposed by the license cause to customer t.

It is worth noting that r, the parameter about license restrictiveness, has a double effect on U_{OS}. A more restrictive license restricts the possible uses of the software and, consequently, the revenues that the embedder can earn from it; this is the direct effect of a more restrictive license on U_{OS}, which we account for with the term tr. In addition, there is also an indirect negative effect that impacts all the OS adopters through the term $\theta(r)$: as long as a larger r places more constraints on the possible uses of the code, OS adopters are less

adoption would not significantly change our results. This way of modelling the timing of adoption decision widely accepted in the literature on technology adoption; among others, see Katz and Shapiro (1986).

[12]The level of t depends both on the nature of the software and on the use that customers make of the software itself. Since customers use the code as an input to produce other, derived, software that they either sell directly or that they embed into their own products, then t is larger when the derived software represents the core of the customers' products/technologies: the more relevant the derived software in the embedded system, the larger the damage for the embedder if forced by the license to release the code under reciprocal licensing terms.

motivated in spending time and effort in improving the code/fixing bugs. In other terms, a larger r affects negatively the extent of the development externality; formally we assume that $\theta(r)$ is a (weakly) decreasing function of r: $\theta'(r) \leq 0$.[13]

The adoption of the proprietary version delivers a net benefit of:

$$U_P(p, r) = V + \theta(r)N - p,$$

that is, the overall quality of the software, $V + \theta(r)N$, net of the price p charged by the firm. It deserves to be stressed that, in this case, the adopter does not receive the disutility from license restrictiveness: by definition, the proprietary version is licensed according to the embedders' preferences; formally, r is set equal to zero in this case. This fact implies that, irrespectively of t, all customers receive the same level of utility from the proprietary version of the software.

Finally, notice that when the firm decides to develop the software completely in-house then only the proprietary version of the software is available to customers. In this case, $\theta(r)N = 0$ and the net utility from adopting the software is simply $V - p$.

For the sake of simplicity, all through the paper we normalize the firm's costs to zero and we assume that customers have mass 1. Furthermore, in order to make the problem of interest, we assume that $V > u_o$; clearly, if $V \leq u_o$, then the strategy of completing the project without releasing the code to the OS community would never be profitable.[14]

2.1 The optimal strategy

Whether going open source is an optimal strategy can be verified by comparing the profits that the firm achieves by distributing only the proprietary version with those obtained under dual licensing. However, the problem can be reduced to the analysis of dual licensing only, once noted that releasing the code to the OS community at an extremely restrictive license is equivalent to sell the proprietary version only; in fact, when the firm sets the degree of

[13]The assumption that $\theta(r)$ is a decreasing function is supported by the empirical literature on OS software, which finds evidence that the level of engagement of the OS community tends to decrease with the restrictions imposed by the license. See among others, Comino, Manenti, and Parisi (2007) and Fershtman and Gandal (2007).

[14]Absent the externality, the utility from adopting the proprietary version is $V - p$, and none would be willing to pay a positive price for the software when $V \leq u_o$.

license restrictiveness to infinity, none would be willing to adopt the OS version and this makes, de facto, the firm distributing only the proprietary version of the software. Formally, we interpret the choice $r \to \infty$ as the case where the firm chooses not to make the open source version of the software available to customers.

In order to characterize the behavior of the firm, we first need to determine how many customers adopt the open source version of the software (i.e. the size of the open source community) and how many pay for the proprietary one. By comparing $U_P(p,r)$ and $U_{OS}(t,r)$, it follows that the customer who is indifferent between the two versions is located at $t = \frac{p}{r}$.

In order to sell to a positive amount of customers, the firm needs to set p and r such that *i)* $U_P(p,r) \geq U_{OS}(t,r)$, for at least some t, and *ii)* $U_P(p,r) \geq u_o$.

Condition *i)* implies that the firm sets the price and the license restrictiveness in a way such that such that $\frac{p}{r} < T$. In turn, condition *ii)* implies that the market is fully covered; customers with $t \geq \frac{p}{r}$ adopt the proprietary version since, in this way, they obtain a net utility larger than both U_{OS} and u_o. Similarly, customers with $t < \frac{p}{r}$ adopt the open source version of the software since this guarantees a net benefit greater than both U_P and u_o.

According to these considerations, the mass of OS adopters is simply given by $N = F\left(\frac{p}{r}\right)$, while the mass of those who buy the proprietary version is equal to $1 - F(\frac{p}{r})$.

The following lemma characterizes a further condition that the pair (p,r) chosen by the firm must satisfy.

Lemma 1. *The firm sets* (p,r) *such that* $V + \theta(r)F\left(\frac{p}{r}\right) - p = u_o$.

Proof. As observed above, customers have homogeneous preferences with respect to the proprietary version of the software. This fact implies that the indifferent customer and all those who adopt the proprietary version of the software obtain the same level of utility. We prove the Lemma by contradiction; let us assume that, at the equilibrium pair (p,r), this common level of utility is strictly larger than u_o, namely $U_P(p,r) = V + \theta(r)F\left(\frac{p}{r}\right) - p = U_{OS}(t = \frac{p}{r}, r) = V + \theta(r)F\left(\frac{p}{r}\right) - \frac{p}{r}r > u_o$. It is easy to show that the firm can do better by increasing marginally, and in the same proportion, p and r up, respectively, to $p' = p(1+\varepsilon)$ and $r' = r(1+\varepsilon)$, where $\varepsilon > 0$ is a negligible number. At the new pair (p', r'), the indifferent customer is still located at p/r; moreover, given that ε is negligible, then $U_P(p', r')$ and $U_{OS}(t = \frac{p}{r}, r')$ are still greater or equal than u_o. Therefore, the masses of customers adopting the two versions of the software do not change with respect to those at the original pair.

Nonetheless, since the firm is selling at a higher price it certainly makes larger profits, thus contradicting the initial assumption. □

The intuition for the above lemma is simple. As in a standard monopoly model with unit demand, the firm optimally sets the price and the restrictiveness of the OS license in order to extract all the surplus obtained by those who adopt the proprietary version of the software. Lemma 1 is important since it implicitly defines the optimal price for the proprietary version as a function of the degree of licence restrictiveness chosen by the firm when posting the source code on a public repository. We define as $p\,(r)$ the price which is implicitly determined by the optimum condition $V + \theta(r)F\left(\frac{p}{r}\right) - p = u_o$.

Thanks to Lemma 1, the firms' optimal strategy can be reduced to the choice of r, and the maximization problem can be simplified as follows:

$$\max_{r \geq 0} \qquad \pi(r) = p(r)\left(1 - F\left(\frac{p(r)}{r}\right)\right).$$

Notice that if the firm would release the OS version without restrictions, then everybody would adopt it: if $r = 0$, then $F\left(\frac{p(r)}{r}\right) = 1$ and the firm makes zero profits since it sells nothing. In the opposite scenario, when r goes to infinity then no one would be willing to adopt the OS version of the software, and $F\left(\frac{p(r)}{r}\right) = 0$; from Lemma 1, the firm sets $p = V - u_o$ and it makes profits equal to $V - u_o$. Formally: $\lim_{r\to\infty} \pi(r) = V - u_o$.

We are now in the position to state the main result of our paper.

Proposition 1. *When the strength of the development externality is sufficiently large, then it is optimal to employ a dual licensing strategy. Formally, when* $\lim_{r\to\infty} \theta(r) > V - u_o$*, dual licensing is profitable.*

Proof. Consider the derivative of the profit function with respect to r. Simple calculations show that $\pi'(r) = p'(r)\left(1 - F\left(\frac{p(r)}{r}\right)\right) - p(r)f\left(\frac{p(r)}{r}\right)\frac{p'(r)r - p(r)}{r^2}$, where, $f(t) = \frac{dF(t)}{dt}$. Using the condition provided in Lemma 1, from the implicit function theorem we have that:

$$p'(r) = \frac{\theta(r)f\left(\frac{p}{r}\right)p - r^2\theta'(r)F\left(\frac{p}{r}\right)}{r\left(\theta(r)f\left(\frac{p}{r}\right) - r\right)}.$$

Using expression $p'(r)$ and Lemma 1, the derivative $\pi'(r)$ becomes:[15]

[15]In order to avoid cumbersome notation, in what follows we omit the arguments of functions $\theta(r)$, $\theta'(r), F(t)$, and $f(t)$.

$$\pi'(r) = \underbrace{f\frac{V-u_o+\theta F}{r}\frac{-\theta+2\theta F+V-u_o}{-\theta f+r}}_{(A)} + \underbrace{\frac{F\left(r\left(1-F\right)-f\left(V-u_o+\theta F\right)\right)}{-\theta f+r}\theta'}_{(B)} \quad (1)$$

Consider $\lim_{r\to\infty}\pi'(r)$; notice that since $V-u_o$ represents the horizontal asymptote of $\pi(r)$, then $\lim_{r\to\infty}\pi'(r)=0$. In order to prove the proposition, we need simply to show that $\pi(r)$ converges to $V-u_o$ from above, namely that $\lim_{r\to\infty}\pi'(r)=0^-$.

Consider term (B) of the above expression; note that as r goes to infinity, then, since $\theta'\leq 0$, this term is negative. Term (A) converges to 0^- provided that $\lim_{r\to\infty}\theta\left(r\right)>V-u_o$. This is enough to prove that there exist (at least) one finite value of r such that $\pi(r)>V-u_o$. □

This result is intuitive. The benefit of dual licensing accrues from the development externality: the contributions of the OS community improve the quality of the code and allow the firm to charge a larger price for the proprietary version. On the other hand, dual licensing entails the risk of cannibalizing the market since the open source version of the software competes with the proprietary one. Dual licensing is a profitable strategy for the firm, whenever the strength of the development externality, $\theta(r)$, is large relative to the quality of the software that the firm is able to develop in-house, V.[16] In particular, the proposition shows that when the strength of the externality is sufficiently large, the firm finds it optimal to set r at a sufficiently low level so that there exists a positive mass of open source adopters.

Proposition 1 provides a general message; in order to characterize more closely the optimal dual licensing strategy, we need to make assumptions on the shape of the externality; the next Proposition presents the optimal strategy chosen by the firm when the strength of the externality is constant, formally when $\theta(r)=\theta$ for all r.

Proposition 2. *Suppose that $\theta(r)=\theta$ for all r, then the firm optimally employs a dual licensing strategy when $\theta>V-u_o$. Under dual licensing, the firm sets a price $\tilde{p}$ and a degree of license restrictiveness $\tilde{r}$ such that, regardless of the c.d.f. $F(t)$, the size of the OS community is equal to $N(\tilde{p},\tilde{r})=\frac{1}{2}\frac{\theta-V+u_o}{\theta}$.*

[16] Since $\theta'(r)\leq 0$, the condition for which the Proposition holds, $\lim_{r\to\infty}\theta(r)>V-u_o$, implies that $\theta(r)>V-u_o$ for any value of r.

Proof. When $\theta(r) = \theta$, expression (1) reduces to:

$$\pi'(r) = f\frac{V - u_o + \theta F}{r}\frac{-\theta + 2\theta F + V - u_o}{-\theta f + r}.$$

The first order condition $\pi'(r) = 0$ is uniquely solved when $F\left(\frac{p(r)}{r}\right) = \frac{1}{2}\frac{\theta - V + u_o}{\theta}$. This implies that when $\theta \leq V - u_o$, the optimal size of the open source community is zero, while when $\theta > V - u_o$, the mass of OS community is equal to $\frac{1}{2}\frac{\theta - V + u_o}{\theta}$. This latter case necessarily identifies a maximum provided that $\pi(0) = 0$ and $\lim_{r\to\infty} \pi(r) = V - u_o$. □

We have already discussed when commenting Proposition 1 that the decision to employ a dual licensing strategy relies entirely on the comparison between the strength of the externality and the in-house quality of the code. Proposition 2 highlights an additional interesting feature of the equilibrium strategy: when $\theta(r)$ is constant, the firm sets p and r in a way such that the mass of OS adopters is independent of the distribution of customers' preferences towards license restrictiveness.

In order to interpret this latter result, it is useful to consider the maximization problem in terms of N rather than r, provided that there is a one-to-one mapping between r and N: $N = F(p/r)$ or $r = p/F^{-1}(N)$. In this respect, Lemma 1 implicitly defines the optimal price as a function of N, formally $p(N) = V + \theta(p(N)/F^{-1}(N))N - u_o$. The firm chooses N to maximize its profits $p(N)(1 - N)$. When θ is constant, an increase in N has two effects: on the one side, firms' profits go up by $\theta(1 - N)$; on the other side, a larger N, reduces sales and, therefore, profits of an amount equal to $p(N) = V + \theta N - u_o$. These two effects do not depend on the function $F(t)$, thus explaining why at the equilibrium the optimal size of the OS community is independent of the distribution of customers' preferences towards r.

Things become more articulated when the strength of the externality decreases with the degree of license restrictiveness. In this case an additional effect must be taken into account by the firm when deciding r: a less restrictive license is generally more desirable since it increases the strength of the development externality. We highlight these arguments in Proposition 3.

Proposition 3. *Suppose that $\theta'(r) < 0$; whenever the firm finds it optimal to endorse a dual licensing strategy, it does so by setting $r < \tilde{r}$, where $\tilde{r}$ is defined in Proposition 2.*

Proof. When $F = \frac{1}{2}\frac{\theta(r) - V + u_o}{\theta(r)}$, $\pi'(r)$ calculated in the proof of Proposition 1 becomes:

$$\frac{(\theta(r) - V + u_o)(V + \theta(r) - u_o)}{4\theta(r)^2}\theta'(r).$$

It is immediate to see that this expression is negative. In fact, $\theta(r) - V + u_o > 0$ when dual licensing is profitable, i.e. when $\lim_{r\to\infty} \theta(r) > V - u_o$. Similarly, given that $V > u_o$ by hypothesis, also $V + \theta(r) - u_o$ is positive. Finally, we are discussing the scenario with $\theta'(r) < 0$. This is enough to prove the Proposition. □

This proposition is interesting and suggests that when $\theta'(r) < 0$, the firm tends to endorse a more "pro OS" strategy by releasing the code to the community under a less restrictive terms; this is a good thing for the firm since it stimulates the contributions from the OS developers, which translates into higher quality of the proprietary version and therefore larger profits.

We conclude this section with a final observation related to the welfare effects of dual licensing:

Remark 1. When the firm employs a dual licensing strategy, it induces a Pareto improvement.

Obviously, whenever the firm chooses to dual licence its code, it does so because it obtains larger profit. But also customers may be better off: without dual licensing, all individuals end up with their reservation utility u_o. With dual licensing, those that purchase the proprietary version still obtain u_o; however, customers adopting the OS version of the software obtain a utility which is strictly larger than the reservation level. The decision to provide the code to the OS community generates some value through the development externality; part of this value goes to the firm and part to individuals, thus explaining the Remark.

2.2 A specific example

To better grasp the intuition behind our model, the following corollary characterizes the optimal strategy when $\theta(r)$ is constant and t is uniformly distributed over $(0, b)$.

Corollary 1. *When $t \sim U(0, b)$ and $\theta(r) = \theta > V - u_o$, then the firm optimally employs a dual licensing strategy with $\tilde{p} = \frac{V-u_o+\theta}{2}$, and $\tilde{r} = \frac{\theta(V-u_o+\theta)}{b(\theta-V+u_o)}$. License restrictiveness is such that $\frac{\partial\tilde{r}}{\partial V} > 0$, $\frac{\partial\tilde{r}}{\partial b} < 0$, and $\frac{\partial\tilde{r}}{\partial\theta} < 0$ when $\theta \in \left(V - u_o, \left(\sqrt{2}+1\right)(V-u_o)\right)$, while $\frac{\partial\tilde{r}}{\partial\theta} > 0$ when $\theta \geq \left(\sqrt{2}+1\right)(V-u_o)$.*

Proof. The equilibrium price and the level of license restrictiveness are obtained by solving the system of equations $V + \theta F\left(\frac{p}{r}\right) - p = u_o$ and $F\left(\frac{p}{r}\right) = \frac{1}{2}\frac{\theta-V+u_o}{\theta}$ and by using the fact

that, for the case of uninform distribution, $F\left(\frac{p}{r}\right) = \frac{p}{rb}$. The comparative statics is obtained by simply differentiating $\tilde{r}$. □

The positive relationship between $\tilde{r}$ and V can be explained following the same arguments used to discuss Propositions 1, and 2: as V increases, the firm benefits from employing a more "proprietary strategy", i.e. by selecting a more restrictive license.

Consider now the role of b that parameterizes the distribution of customers preferences. From Proposition 2 we know that the firm sets the license restrictiveness in order to optimally segment customers into OS and proprietary adopters. When b gets larger the mass of customers that are substantially affected by the license restrictiveness increases; therefore, the firm needs to reduce r in order to enlarge the mass of OS adopters up to the optimal size defined in Proposition 2.

The impact of an increase in the strength of the development externality on $\tilde{r}$ is more articulated and it entails to two opposite effects. A larger value of θ signals that the contribution of the OS community is highly valuable. Nonetheless, a larger θ makes the open source version of the software also a stronger competitor vis a vis the proprietary one; more specifically, as θ increases a larger share of customers is attracted by the OS version of the product. The former effect dominates whenever the size of the OS community is relatively small, that is when the strength of the externality is not too large, $\theta \in \left(V - u_o, \left(\sqrt{2}+1\right)(V - u_o)\right)$. In this case, the firm benefits from augmenting the size of the community through a reduction in the level of license restrictiveness. On the opposite, the "competition effect" prevails when the OS community is already sufficiently large; in this case, the firm reacts to a further increase in θ by selecting a larger $\tilde{r}$.

3 Discussion and future research

In this paper we have proposed a theoretical model to study the characteristics and the commercial sustainability of a particular open source business strategy known as dual licensing. The focus is on the decision of a software vendor about whether to develop a fully proprietary version of a software or to employ a dual licensing strategy, in a context where customers are commercial firms that are harmed by the restrictions imposed by OS licenses. We have shown that dual licensing is preferred when the feedbacks of the OS community

(the development externality) are valuable enough compared to the quality of the software that the firm is able to develop on its own.

Our analysis points to the crucial role of OS licensing schemes for firms embracing open source strategies. Through an appropriate definition of the licensing terms of distribution of the OS version of the software, the firm balances the opposing effects of going open source. A more restrictive license protects the proprietary version of the software against the risk of cannibalization at the cost of reducing the size of the OS community that contributes to software development; moreover, licensing terms also affect OS programmers' incentives to contribute to the development of a better software.

Even though the theoretical model focuses on the role of reciprocal provisions in making versioning viable, our results have a broader interpretation. As discussed in the Introduction, there are additional dimensions of OS licenses that might disturb potential customers;[17] in these cases, a software house may profitably go OS and sell an "upgraded" version of the software to those customers who are willing to pay to be freed from the specific provisions/limitations of the OS version

The importance of an appropriate management of OS licenses for software vendors, contributes explaining one of the most debated phenomenon in the OS world, known as "license proliferation".[18] At the time of writing this paper, more than 70 different licensing schemes have been registered as OS licenses; these licenses differ along several dimensions.[19] Interestingly, various commercial vendors have created their own open source license, thus confirming a possible strategic role in the "design" of the license.[20]

[17] See footnote 5.

[18] License proliferation represents one of the major challenges to OS; indeed, the presence of different schemes may pose serious problems given that some licenses are potentially incompatible with each other; for a discussion see Rosen (2004), chapter 10, and the report of the "License Proliferation Committee", available at the Open Source Initiative web-site, `www.opensource.org/proliferation`.

[19] Take, for instance, the reciprocal provision; not all the OS licenses impose such provision (this is the case of the BSD and the other so-called "academic licenses") while, at the same time, an extreme heterogeneity in terms of the degree of reciprocity imposed on derivative works can be observed between those licenses that do have reciprocal provisions.

[20] The case of Nokia is emblematic. At the url `opensource.nokia.com/` several different software projects are available for download and often projects are licensed under different terms. Some projects are distributed under the Nokia Open Source License, others are available under different OS licensing templates such as: GPL, BSD, Mozilla Public License, LGPL, and others. Similarly, also IBM, Intel and Microsoft have created

One simplifying assumption that we have implicitly made in the paper and that deserves further discussion is that the only way for the firm to benefit from the contribution of the community is by making the OS version of the code freely available on a public repository. This assumption is made on practical grounds; the fact that a vast majority of OS projects hosted on public repositories such as SourceForge.net and sponsored by commercial vendors are freely available, goes exactly in this direction. More specifically, this assumption is supported by the observation of the strategies adopted by those firms, such as Oracle and Sun Microsystems, whose experiences have been inspiring our paper. This assumption, however, is not innocuous since it implies that the firm cannot do better by following other strategies, more articulated than those described in the paper. For example, the firm cannot benefit from selling at a positive price an open source version of the software (i.e. a version with $r > 0$). Indeed, in our framework, adopters of this version of the software are assumed not to contribute to the development of the project; moreover, since they bear the disutility due to the restrictions imposed by the OS license, they are willing to pay a price which is smaller than what they would pay for the proprietary version. Consequently, the firm is certainly better-off inducing these customers to adopt either the freely available OS version (to let them contribute to the project), or the proprietary version (to charge them a larger price). Similar arguments apply to the feasibility of multiple licensing strategies, i.e. the release of more than one OS version of the software at a zero price; in this case, OS adopters will certainly select the version released under the less restrictive license.

Finally, in the paper we focus on the behavior of a monopolist producing a certain software. An important extension that we leave for future research relates to the role of open source strategies in competitive frameworks.

their OS license.

References

Belleflamme, P. (2005). Versioning in the Information Economy: Theory and Applications. *CESifo Economic Studies*, 51:329–358.

Comino, S., Manenti, F. M., and Parisi, M. L. (2007). From Planning to Mature: On the Success of Open Source Projects. *Research Policy*, 36:1575–1586.

Daffara, C. (2009). FLOSSMETRICS: The SME guide to open source software. Document available at http://flossmetrics.org/.

Economides, N. (1996). Network Externalities, Complementarities, and Invitations to Enter. *European Journal of Political Economy*, 12:211–233.

Fershtman, C. and Gandal, N. (2007). Open source software: Motivation and restrictive licensing. *International Economics and Economic Policy*, 4(2):209–225.

Gayer, A. and Shy, O. (2003). Internet and Peer-to-Peer Distributions in Markets for Digital Products. *Economic Letters*, 81:51–57.

Jullien, N. (2006). New Economic Models, New Software Industry Economy. RNTL Report.

Katz, M. and Shapiro, C. (1986). Technology Adoption in the Presence of Network Externalities. *Journal of Political Economy*, 94(4):822–841.

Lanzi, D. (2009). Competition and Open Source with Perfect Software Compatibility. *Information Economics and Policy*, 21:192–200.

Moody, G. (2006). Does Dual Licensing Threaten Free Software? *Linux Journal*, page 27^{th} July.

Mueller, M. (2007). OpenOffice.org Projects by Members. `blogs.sun.com/GullFOSS/entry/openoffice_org_projects_by_members`.

Mustonen, M. (2005). When Does a Firm Support Substitute Open Source Programming? *Journal of Economics & Management Strategy*, 14(1):121–139.

Rajala, R., Nissilä, J., and Westerlund, M. (2007). *Revenue Models in the Open Source Software Business*. In Kirk St. Amant and Brian Still eds. Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives, IGI Global.

Rosen, L. (2004). *Open Source Licensing. Software Freedom and Intellectual Property Law*. Prentice Hall.

Seigo, A. (2006). The Quest for Project Identity and Definition. Keynote Speech Akademy Conference.

Shapiro, C. and Varian, H. (1998). Versioning: The Smart Way to Sell Information. *Harvard Business Review*, 76(6):106–114.

Välimäki, M. (2005). *The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software Industry*. Turre Publishing.

West, J. and Gallagher, S. (2004). Key Challenges of Open Innovation: Lessons from Open Source Software. San José State University, mimeo; document avaiable at www.joelwest.org/Papers/WestGallagher2004.pdf.