Dual Licensing in Open Source Software Markets

Stefano Comino†  Fabio M. Manenti‡

November 2008

Abstract

In this paper we present a theoretical model to study the characteristics and the commercial sustainability of a particular open source (OS) strategy known as dual licensing, the practice of distributing a software both under a commercial and an OS license. We show that the decision to employ a dual licensing strategy occurs whenever the strength and the relevance of the contribution of the OS community are sufficiently large, while it does not depend upon customers’ preferences towards license restrictiveness. The profitability of dual licensing crucially depends on the proper setting of the licensing terms of the OS distribution, which represents the key versioning tool; this analysis suggests a possible explanation for the observed proliferation of open source licenses.

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

Keywords: open source software, 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 2nd FLOSS international workshop on Free/Libre open source software - Rennes 2008, and at the 3rd annual conference of the EPIP association - Bern, 2008. 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 (Hecker, 1999; Bonaccorsi et al., 2006; Rajala et al., 2007).

Commercial firms may enjoy several benefits by “going open source”. For instance, 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. 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. Also, the decision to release the source code to the OS community might be necessary in order to cope with an increased competition in the marketplace.\(^1\)

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.\(^2\) A typical commercial strategy that has been successfully employed is to sell complementary products and to profit from OS-related segments and/or services.\(^3\)

In this paper we focus on an alternative business model that is gaining popularity among software vendors which is known as dual licensing.\(^4\) With dual licensing firms mix traditional

\(^1\)According to many commentators, Netscape started the Mozilla project due to its inability to meet the competitive threat posed by Internet Explorer; see West and Gallagher (2004).

\(^2\)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.

\(^4\)Besides the case of Sleepycat that has inspired our paper, an ever growing number of commercial firms is employing dual licensing to profit from OS solutions. The most notable ones are certainly MySQL, Trolltech or Red Hat, but also SugarCRM, JasperSoft, or Aladdin Enterprises deserve to be mentioned. See Välimäki...
and OS-based strategies by offering their 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 reasons 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. To better grasp this critical issue, it is useful to quote Michael Olsen, CEO of Sleepycat Software Inc., producer of the embedded database BerkeleyDB; Olsen pioneered the practice of dual licensing, which he describes as follows:

“The Sleepycat 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 Sleepycat Software.” (Välimäki, 2005 pp. 209-210)

Customers that use, modify and embed Berkeley DB into their own applications might be reluctant to use the OS version. These applications may be products per se or, more frequently, they may constitute 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 product, they may be willing to pay in order to be relieved from the reciprocal provision imposed by the open source version.\(^5\) In this paper we consider a profit maximizing firm that is developing a software project targeted to commercial customers. The firm can either develop the project completely in-house or it can employ a dual licensing strategy. In this latter case, the firm needs to balance two opposing effects: on the one side, it may

\(^5\)It is well known that commercial firms may be really unwilling to embed OS code into their own proprietary products; one may think, for example, to the fears of the many companies that embed OS Linux into their own platforms but that, despite the license requirements, decide not to release their derived code under the GPL because it represents the bulk of their product’s core technology. They fear that some day a court may oblige them to adhere to the GPL reciprocal provision and to release their derived software to the public domain. This fear has been exacerbated since the well known “SCO vs IBM” case, whereby IBM has been alleged to violating SCO’s intellectual property rights by distributing a Linux distribution with copied code; see Moglen (2003).
benefit from the contributions of the OS community, allowing for a better quality of the product; on the other side, the firm faces the risk of cannibalization since the provision of the OS version will inevitably shrink the market for the proprietary one. As we show in the paper, the managing of the open source license plays a crucial role in balancing pros and cons of a dual licensing strategy. Assuming that customers have heterogenous preferences towards the restrictions imposed by the OS license, 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. Interestingly, we show that the decision of whether to employ dual licensing or not, i.e. the choice between developing the software completely in-house or resorting to the help of the OS community, does not depend on the distribution of the customers’ preferences towards the restrictions imposed by the OS license; eventually, the key “versioning” tool is represented by the terms of distribution of the OS version: the firm can adapt its OS strategy to the characteristics of the customer preferences through an appropriate “micromanagement” of the licensing terms.

According to this view, dual licensing is a typical example of versioning (Belleflamme, 2005; Shapiro and Varian, 1998); more generally, the relevant message of this paper is that the definition of the licensing terms represents an important strategic tool for those commercial firms that want to profit from OS solutions.

Our result also suggests a possible, although not certainly unique, explanation for one of the most debated phenomenon in the OS world, known as “license proliferation”. 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. Take, for instance, the reciprocal provision: not all the OS licenses impose such provision, 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. Interestingly, various commercial vendors have created their own open source license, thus suggesting a possible

---

6License 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.

7The most notable examples are represented by the BSD (Berkeley software distribution) and by the other so-called “academic licenses” that do not impose reciprocity.
strategic role in the “design” of the license. This is the case of IBM, Intel, Microsoft, or Nokia, just to mention some notable examples.\footnote{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.}

Apart from the reciprocal provision, there are other reasons that might render the proprietary version of the software preferred to the open source one. Customers may be interested in obtaining products with more warranties or less limitations of the liability of the licensor than those usually available for OS licensed products; or they might want to receive technical or aftersale support from the software vendor which is not normally offered to those that adopt the open source version. Also, in many instances, OS licenses allow the licensor to terminate the agreement conditional on the occurrence of specific events,\footnote{This is the case of, for instance, the so-called “patent termination clauses”; for a discussion of this issue see Rosen (2004).} and this clearly puts the customer to a risk in case she/he needs to invest money and effort in using the software. Finally, the proprietary version may come with additional features and functionalities that are of particular value to customers.

The paper is organized as follows: the model is presented in Section 2 and in Section 3 we conclude.

\section{The model}

Suppose that a commercial firm has started developing a new software which is directed to other commercial firms that need the source code of the software to embed it into their own products. These customers are interested not only in the quality of the code “per-se”, but also to the terms of licensing under which it is distributed: they want to keep full control on the products they develop, and prefer to obtain the code under unrestrictive licenses.

The firm may follow two alternative strategies to develop the software: it can either complete the project on its own, or it can post the source code in a public repository in order to benefit from the contributions of the open source community, thus enjoying of the so called “development externality”;\footnote{Our model does not consider standard network effects based on usage; this type of externality is not} in the latter case we say that the firm is employing a
dual licensing strategy in that its product is available both under an OS as well as under a proprietary license.

The firm takes its decisions sequentially; the timing of the game is as follows:

1. the firm decides whether to release the source code to the open source community; if it releases the code, the firm sets the degree of restrictiveness of the OS licence: formally, it chooses \( r \in [0, \infty) \);

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

Customers observe the firm’s licensing strategy and take their adoption decision. 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.

Customers may either adopt the proprietary version or, whenever available, the OS one; in the former case, a customer obtains a net benefit equal to

\[
U_P(p) = V + \theta N - p,
\]

where \( V + \theta N \) is the overall quality of the software, with \( V \) representing the contribution of the firm and \( \theta N \) the development externality accruing from the OS community, which is increasing in \( N \), the mass of open source adopters, and in the strength of the externality, \( \theta > 0 \). Alternatively, whenever the firm has released the open source version of its product, a customer can download the software posted on the repository without paying any licensing relevant in our framework since at the equilibrium, also in case the software is kept proprietary, the market is fully covered and, irrespectively of the strategy adopted by the firm, the installed base of users is always equal to 1. Furthermore, as shown in Bonaccorsi and Rossi (2004) in a survey based on a sample of 146 Italian software firms, the development externality is the main motivation for going open source.

\(^{11}\)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 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).
fee;\footnote{Note that once the code has been released to the OS community, competition may take other forms too. OS programmers might download the code and start independent development on it. In the industry jargon, this phenomenon is known as forking; in the present version of the paper we do not consider the possibility of forking.} given \( r \), the level of license restrictiveness set by the firm, the net benefit enjoyed upon the adoption of the OS version, is

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

where \( tr \) represents the disutility due to the restrictions imposed by the license.\footnote{When it sells the proprietary version, the firm tailors the license to the customers’ needs; therefore, for the proprietary version of the software \( r = 0 \).} 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.

We assume that customers have heterogeneous preferences with respect to license restrictiveness and, in particular, we assume that \( t \geq 0 \) is distributed according to \( F(t) \). All through the paper we will assume that \( F(t) \) is such that the maximization problem of the firm is concave and that the customer indifferent between adopting the OS or the proprietary version is located at an interior point of the support.

\[ \text{2.1 The optimal strategy} \]

In order solve the model, we need to define the customer who is indifferent between the two versions of the software and the size of the OS community as functions of \( p \) and \( r \). By comparing \( U_P(p) \) and \( U_{OS}(t, r) \) defined above, it follows that the customer who is indifferent between adopting the proprietary and the OS version of the software is located at \( t = \frac{p}{r} \). All the individuals located at the left of the indifferent customer will adopt the OS version; therefore the size of the open source community contributing to the development of the project is equal to \( F\left(\frac{p}{r}\right) \).

The following lemma characterizes the condition that the pair \((p, r)\) chosen by the firm must satisfy.
Lemma 1. The optimal \((p, r)\) pair set by the firm is such that \(V + \theta F \left( \frac{p}{r} \right) - p = 0\).

Proof. We proof the result by contradiction. Suppose that \(V + \theta F \left( \frac{p}{r} \right) - p = x > 0\); in this case the firm obtains a profit equal to \((V + \theta F \left( \frac{p}{r} \right) - x) \left(1 - F \left( \frac{p}{r} \right) \right)\). Clearly, the firm can do better by increasing \(p\) and \(r\) in the same proportion, thus keeping the ratio \(\frac{p}{r}\) constant; by doing so, the masses of OS and proprietary adopters do not change, but the price charged to the proprietary adopters is larger. Suppose, on the opposite, that \(V + \theta F \left( \frac{p}{r} \right) - p < 0\); this cannot be an optimal choice either: at this pair \((p, r)\), none of the consumers is willing to buy the proprietary version and the firm makes zero profits. This is enough to prove that the firm’s optimal choice is to set \(p\) and \(r\) such that the condition \(V + \theta F \left( \frac{p}{r} \right) - p = 0\) is satisfied. 

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 defined by the optimum condition \(V + \theta F \left( \frac{p}{r} \right) - p = 0\).

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

Proposition 1. When \(V < \theta\) the firm optimally employs a dual licensing strategy, while it keeps the software proprietary otherwise. Under dual licensing, the price and the degree of license restrictiveness are set such that the size of the OS community is equal to \(\frac{1}{2} \frac{V - \theta}{\theta}\).

Proof. The firm sets \(r\) in order to maximize its profits \(p(r) \left(1 - F \left( \frac{p(r)}{r} \right) \right)\). The optimal \(r\) solves the first order condition w.r.t. \(r\):

\[
p'(r) \left(1 - F \left( \frac{p(r)}{r} \right) \right) - pf \left( \frac{p(r)}{r} \right) \frac{rp'(r) - p(r)}{r^2} = 0,
\]

where \(p'(r) = \frac{dp(r)}{dr}\), and \(f(r) = \frac{dF}{dr}\). Using the condition provided in Lemma 1, from the implicit function theorem we have that \(p'(r) = \frac{\theta f \left( \frac{p(r)}{r} \right) \left(\frac{V + \theta F \left( \frac{p(r)}{r} \right)}{r} \right)}{\theta f \left( \frac{p(r)}{r} \right) - 1}\); plugging this expression into (1) yields:

\[
f \left( \frac{p(r)}{r} \right) \frac{V + \theta F \left( \frac{p(r)}{r} \right)}{r} - \theta + 2\theta F \left( \frac{p(r)}{r} \right) + V
\]

\[
- \theta f \left( \frac{p(r)}{r} \right) + r = 0.
\]
The above equality is satisfied for $F\left(\frac{r(r)}{\theta}\right) = \frac{1}{2} \frac{\theta-V}{\theta}$. This implies that when $\theta \leq V$, the optimal size of the open source community is zero, while it is equal to $\frac{1}{2} \frac{\theta-V}{\theta}$ when $\theta > V$. \qed

Proposition 1 shows that the firm employs a dual licensing strategy whenever the quality of the software that it is able to develop in-house, $V$, is small relative to the strength of the development externality, $\theta$. It is worth noticing that the decision to endorse a dual licensing strategy does not depend on the characteristics of the distribution of the individual preferences with respect to $r$. This fact is interesting since it points to the central role played by OS licenses; by selecting an appropriate $r$ the firm adapts its OS strategy to the specific characteristics of the customers’ preferences, being able to optimally segment adopters into two groups: those that adopt the OS version and contribute to the development of the software, and those that pay for the proprietary version.

To better grasp the intuition on how the firm defines its licensing strategy, the following corollary describes the optimal design of the OS license for the particular case of uniform distribution.

**Corollary 1.** Assume that $t$ is distributed according to $U(\mu - \delta, \mu + \delta)$; when $\theta > V$, the firm employs a dual licensing strategy and sets $r^* = \frac{\theta(\theta + V)}{2(\theta - \delta)(\theta + \delta)}$. License restrictiveness is such that $\frac{\partial r^*}{\partial V} > 0$, $\frac{\partial r^*}{\partial \delta} > 0$, $\frac{\partial r^*}{\partial \mu} < 0$, and $\frac{\partial r^*}{\partial \mu} < 0$ when $\mu > 3\delta$ or when $\mu < 3\delta$ and $\theta \in \left(0, V^2 + (\mu + \delta)^2\right)$, while $\frac{\partial r^*}{\partial \mu} > 0$ otherwise.

**Proof.** The equilibrium price and the level of license restrictiveness are obtained by solving the system of equations $V + \theta F\left(\frac{p}{r}\right) - p = 0$ and $F\left(\frac{p}{r}\right) = \frac{1}{2} \frac{V}{\theta}$ and by using the fact that, for the case of uniform distribution, $F\left(\frac{p}{r}\right) = \frac{1}{2} \frac{\mu - r}{\mu}$. The comparative statics is obtained by simply differentiating the optimal $r$. \qed

The positive relationship between $r^*$ and $V$ can be explained following the same arguments used to discuss Proposition 1: 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 $\delta$ and $\mu$ that determine the characteristics of the distribution of customers. The parameter $\delta$ represents the dispersion of the distribution of customers’ disutility towards $r$ and according to Corollary 1, the firm tends to set a more restrictive license as $\delta$ increases. The intuition for this result stems from what shown in Proposition 1; the firm set the license restrictiveness in order to optimally segment customers into OS
and proprietary adopters. When individuals are highly dispersed, there are customers with a very small $t$, i.e. which are willing to adopt the OS version even if the OS license imposes strong obligations. In this case, the firm can accomplish its objective of stimulating enough OS adoption, even by setting a relatively restrictive license.

The effect of an increase in $\mu$, the expected value of $t$, goes exactly in the opposite direction; as $\mu$ increases, the firm needs to reduce the restrictiveness of the OS license in order to benefit enough from the feedbacks and from the contributions of the open source community.

The impact of an increase in the strength of the development externality on $r^*$ is more articulated and it entails to two opposite effects. A larger value of $\theta$ signals that the contribution of the OS community is highly valuable. Nonetheless, a larger $\theta$ makes the open source version of the software also a stronger competitor vis a vis the proprietary one; more specifically, as $\theta$ 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 either the expected value of $t$ is large relative to its dispersion (formally when $\mu > 3\delta$), or when the strength of the externality is not too large. In both these cases, 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 $\theta$ by selecting a larger $r^*$.

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 strategy known as dual licensing. In a context where customers are assumed to have heterogeneous preferences towards the restrictions imposed by the open source licenses, we have focussed on the decision of a software vendor about whether to develop a fully proprietary version of the software or to employ a dual licensing strategy. We have shown that this decision does not depend on the distribution of customers’ preferences towards the OS licensing restrictions.

This result points to the crucial role of OS licensing schemes for firms embracing open source strategies, and suggests a possible explanation to the observed proliferation of OS
licenses. 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, namely the benefits obtained from the contributions of the OS community on the one side, and the risk of cannibalizing its business on the other.

Even though the paper focuses on dual licensing and on the role of the reciprocal provision, our result has a broader interpretation and it can be extended to a much wider set of open source strategies. As discussed in the introduction, there are various dimensions of an OS license that can disturb potential customers; in these cases, a software house may profitably go OS by setting appropriately the terms of licensing of the OS version, and by selling an “upgraded” version of the software, or the software bundled together with various possible forms of hardware, to those customers who are willing to pay to be freed from the specific provisions/limitations of the OS version, or that are just interested in the bundle.

Our analysis is based on a number of assumptions that, at this point, deserve further discussion. We have implicitly assumed 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 vast majority of projects sponsored by commercial vendors that are freely available on public repositories such as SourceForge.net, goes exactly in this way. More specifically, this assumption is supported by the observation of the strategies adopted by those firms, such as Sleepycat, 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 a 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.
Another crucial assumption is on the strength and the characteristics of the contributions of the open source developers. In our model, the OS license is used by the firm to sort out OS and proprietary adopters: the larger the value of \( r \), the higher the share of proprietary adopters. More realistically, one might think of other possible sorting effects in case the pool of potential adopters is not only characterized by the presence of commercial actors. More idealistically motivated individuals might be willing to contribute to the project only if it is governed by highly restrictive licenses. These arguments drive us straight towards another relevant issue in the OS arena: the “quality” of the contributions, which might depend upon the licensing terms; in other words, the parameter \( \theta \) that describes the strength of the development externality might be a function of the features of the OS license. We have not taken this issue into account not only to keep as simple as possible the algebra of the model, but also, and more relevantly, because the relationship between license type and developer contributions is an ambiguous and debated issue.\(^{14}\)

Finally, the paper just focusses 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.

\(^{14}\)For many reasons, licenses that are more restrictive might encourage OS developers to contribute; career prospects and talent-signaling incentives are generally stronger the more visible the contribution of each developer and this happens to occur when the project is governed by a highly restrictive license (GPL). At the same time, there are also reasons why less restrictive licenses might attract greater involvement; in this direction, there is evidence that projects distributed under highly restrictive licenses are less likely to succeed (Comino et al., 2007; Lerner and Tirole, 2005).
References


