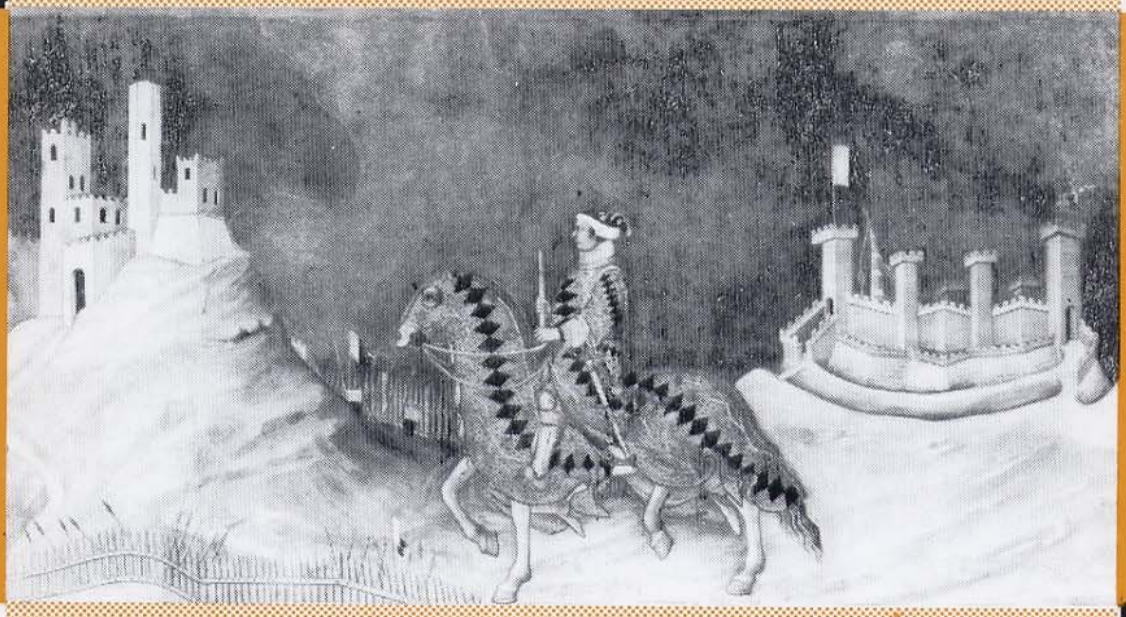
UNIVERSITÀ DEGLI STUDI DI SIENA

QUADERNI DEL DIPARTIMENTO
DI ECONOMIA POLITICA

Massimo D'Antoni
Maria Alessandra Rossi

Copyright vs. Copyleft Licencing and Software Delvelopment

n.510 - Agosto 2007



Abstract - This article aims at clarifying the role played by licenses within the increasingly relevant Open Source Software (OSS) phenomenon. In particular, the article explores from a theoretical point of view the comparative properties of the two main categories of OSS license--copyleft and non-copyleft licenses--in terms of their ability to stimulate innovation and coordination of development efforts. In order to do so, the paper relies on an incomplete contracting model. The model shows that, in spite of the fact that copyleft licenses entail the enjoyment of a narrower set of rights by both licensors and licensees, they may be preferred to non-copyleft licenses when coordination of complementary investments in development is important. It thus provides a non-ideologically-based explanation for the puzzling evidence showing the dominance, in terms of diffusion, of copyleft licenses.

Keywords: intellectual property rights, open source, copyright, copyleft, GPL license, incentives to innovation.

JEL classification: L17, O34.

We thank Simone Piccardi for his help and encouragement, and for providing invaluable information on the open source world, and Sam Bowles for his comments and suggestions. However, all errors remain ours.

Massimo D'Antoni, Dipartimento di Economia Politica, Università di Siena
Maria Alessandra Rossi, Dipartimento di Economia Politica, Università di Siena

1. Introduction

Open Source Software (OSS) has reached a significant extent of market penetration in recent years. A June 2006 report by research firm Gartner suggested that OSS would take away 22% of the traditional software market over the next five years. In July 2006 IDC estimated that OSS held 7% of IT software revenue and projected an increase to 15% of IT software budgets in the next four years. Considering that a significant part of OSS products is distributed for free, the latter projection may well underestimate the extent of OSS diffusion. Moreover, OSS is the market leader in the web server segment, where Apache holds about 54% of the market according to the latest Netcraft survey (June 2007) and holds relevant market shares in the mail server market (47,8%, according to the FalkoTimme mail server survey) and in the database market (33% of European firms use OSS databases, according to IDC).

Economic scholarship has kept up with the pace of OSS market diffusion, exploring a wide range of OSS-related issues and reconciling many apparently puzzling characteristics with conventional economic analysis (for a survey, see Rossi, 2006). Yet, a few relevant issues—and particularly the role played by licenses—remain under-researched. Indeed, the search for economics-based explanations for the OSS phenomenon has led to the identification of features, such as the interplay between intrinsic and extrinsic motivations to contribute, that may make appear the very existence of licenses redundant, so that OSS software is often assimilated to software in the public domain.

However, although OSS software is usually freely available to anyone who cares about making use of it, differently from software in the public domain it is protected by copyright and distributed under various sorts of licenses that may set restrictions to its redistribution. The rationale for the existence of various sorts of licenses is also still relatively unclear. Indeed, OSS licenses vary greatly as regards the amount of restrictions they impose on licensees. In particular, so-called copyleft licenses grant developers a narrower set of rights with respect to non-copyleft licenses and thus dramatically reduce their ability to profit from the direct sale of the software and make more difficult the combined commercialization of OSS and proprietary software programs.

In this regard Maurer and Scotchmer (2006), for instance, note that:

> "[T]he need for licenses is not entirely obvious nor, assuming that licenses are needed, is it clear which restrictions are necessary or desirable. From a welfare standpoint, the best way to ensure use and re-use of software would be to place it in the public domain

without any license at all." (p. 17)

Thus, it is not entirely clear whether licenses play a role in ensuring the viability of OSS nor is it clear whether different types of OSS licenses do have different implications for the pace and dynamics of development.

In this paper, we aim to contribute to the understanding of the OSS phenomenon by focusing on the role licenses play within OSS projects, and particularly on the comparative properties of the copyleft and non-copyleft licenses (as exemplified respectively by the GPL and the BSD license, the most popular in their classes). We build a formal model that takes as a starting point the recognition of the specific nature of the investments in software development–an aspect too often overlooked–and the associated ex-ante inefficiencies in the investment choices arising as a consequence of contractual incompleteness. The model highlights the implications in terms of incentives to invest in software development of the alternative property rights allocations realized through different OSS licenses.

The model shows the (perhaps counterintuitive) result that, in spite of the fact that copyleft licenses impose more stringent restrictions on both licensors' and licensees' residual rights of control relative to non-copyleft licenses, they may induce higher levels of coordination of investment and thus be preferred to non-copyleft licenses in order to enhance the degree of co-specificity of the investment efforts. In so doing, it allows both to shed light on a number of empirical observations that have so far escaped theoretical scrutiny and to provide insights on the implications of different OSS licensing choices for firms.

The incomplete contracting framework we adopt is loosely related to the "GHM approach", namely the collection of contributions by Sanford Grossman, Oliver Hart and John Moore (Grossman and Hart, 1986; Hart and Moore, 1990; Hart, 1995), and to the other few contributions that have applied some insights from the GHM approach to the analysis of issues arising in innovative contexts (see, e.g. Aghion and Tirole, 1994; Arora and Merges, 2001). Differently from the GHM approach and from the other mentioned contributions, however, we introduce an additional dimension to the choice of investments by agents. While in GHM-style models agents choose only the level or intensity of specific investments, agents in our model may choose both the intensity and the degree of complementarity/specificity of investments[1]. Adding this further dimension is important because our focus is on contexts of cumulative innovation in which it is important to assess not only the intensity of incentives to invest but also the extent of coordination

[1] In this respect, our approach can remind of Cai (2003), where the choice of the degree of specificity is used to justify forms of common property.

of investment. This, in turn, implies that in evaluating the effects of different licenses their impact on both of these dimensions should be taken into account.

The paper is organized as follows. Section 2 explores the very rationale of the choice of opening the source code. Section 3 describes the principal types of OSS licenses, introducing the difference between copyleft and non-copyleft licenses. Section 4 presents a formal model that captures the main elements of the comparison between copyleft and non-copyleft licenses. Section 5 discusses the mainresults of the model, introduces some stylized facts on which the model may shed light, and concludes.

2. The choice of opening the source code

The defining characteristics of OSS are (a) the free availability of its source code, i.e. of the human-readable instructions expressing the different tasks that have to be performed by the computer, and (b) the nature of the license under which it is distributed, which grants licensees a number of rights, namely the right to use (run) the program, to study how it works, to modify and improve it, to redistribute it with or without modifications[2]. Of course, the first condition (free access to the source code) is a precondition for the second in that no improvement is possible in absence of access to the source code[3].

It is important to note that the choice to release a piece of software under an OSS license does not involve giving up the copyright over it. This

[2] The free software definition makes explicit reference to the large set of rights (freedoms) accorded by OSS licenses:

> Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedoms, for the users of the software: The freedom to run the program, for any purpose (freedom 0). The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbor (freedom 2). The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this. A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.

[3] Note that OS software is to be distinguished from software whose licence allows to use it freely, but not to modify it (e.g. Acrobat). In this case the software is free, in the sense that is it distributed at no cost, but it is not open source.

distinguishes the choice to distribute the software under an OSS license from the choice to release it in the public domain[4]. The release of the software in the public domain entails that the consent of the author of the software is no longer required for third parties to use and modify it. The same result is achieved by OSS licenses through contractual means, rather than through renouncing to property altogether. However, differently from public domain software, by using an OSS license the licensor may impose specific restrictions to some aspects of the redistribution of the software. This will become clearer in what follows and will play an important role in explaining the comparative properties of different types of OSS licenses.

Of course, in any case opening the source code reduces the extent to which the developer of the software licensed under OSS terms (or any other software developer) may profit from the *direct sale* of it, although important differences exist in this regard in relation to the specific type of OSS license chosen (on which more will be said in the following section). The question therefore arises of why a rational individual would ever choose to release her software under an OSS license. This question—a "puzzle" for many—has been for long prominent in the literature on OSS. Answers range from the identification of the reputational and signalling benefits of contributing to OSS projects (Lerner and Tirole, 2002), to the recognition of benefits in terms of satisfaction of specific user needs (von Hippel, 2002; Johnson, 2002), to the pinpointing of various sorts of intrinsic motivations, including the enjoyment of programming *per se* (Moglen, 1999) and an ideological commitment to the norms of OSS communities and the very idea that source code should be open (Raymond, 1998; Bergquist and Ljungberg, 2001).

In this paper, we disregard ideological explanations for the choice of an OSS license by focusing on the incentives of profit-oriented individuals or firms, interested in maximizing the value of the software they work with[5]. For this sort of agents, opening the source code is crucial to enable investment by multiple parties with heterogeneous human capital. Indeed, the combination of free access to the source code and the wide scope of rights over the licensed software creates an opportunity for multiple agents to have simultaneous access to the same software program and eventually invest in

[4] In order to release his own work in the public domain, the author of a piece of software must take some explicit legal steps in order to disclaim the copyright over it, given that copyright immediately attaches to original creations under the Berne Convention for the Protection of Literary and Artistic Works of 1886.

[5] Note that this does not mean taking any particular stance on the comparative relevance of different motivations for releasing software under OSS licenses. In other words, we do not rule out the possibility that in many cases intrinsic motivations may play an important role.

its development. This is particularly relevant in light of two characteristics of software development: cumulativeness and investment specificity. These characteristics imply that, although an OSS strategy reduces the profits from the direct sale of the software, it might entail greater benefits than costs.

Consider the two aspects in turn. First, an OSS strategy entails important technical benefits in a context of cumulative innovation such as software development. Software innovation is strongly incremental, i.e. it results from a cumulative process in which improvements build on previous versions and developers rely both on their own and on others' existing designs and examples in order to incorporate them into new programs or adapt them to serve new purposes.

Cumulativeness implies that a given software constitutes the input into further development efforts. In this context, it is technically possible to independently develop two programs or software modules meant to be used jointly without having access to the source code as long as some instructions on the realization of interfaces are provided by the licensor of the original software input. However, in keeping the source code secret, important gains in technical efficiency are foregone. By opening the source code, by contrast, improvements and complementary programs can be developed in a way that increases the value of joint use of the software and optimizes the interaction between the different programs/modules. In particular, only if the source code of a given software is open it is possible to coordinate the development efforts of different agents operating in a decentralized fashion (i.e. outside of the boundaries of a firm). In addition to this, access to the source code can bring about as a side effect improvements in the form of bugs or error corrections or of the provision of more substantial additions.

Secondly, the existence of an incentive to open the source code may become even more apparent in light of the fact that investments made in the development of a given software program are specific. That investments in software development are specific is a truism too often overlooked in analyses of software innovation that we think is crucial to understand both an initial developer's decision to release a piece of software under OSS terms and the decision by other developers to subsequently contribute to the improvement of that software[6].

Investment specificity implies, on one side, that developers may profit from the sale of complementary services or customized solutions that include

[6] One important exception in this regard in the OSS context is the mentioned 2005 paper by Lerner and Tirole where the authors stress in a footnote that the "hijacking" possible under permissive licensing terms may deprive contributors of some of the benefits from participating to a project because it creates the possibility of hold up and that restrictive licensing terms may be interpreted as a contractual response to this problem.

but do not coincide with the software program[7]. On the other side it implies that, in a context of cumulative innovation, they are interested in having access to the software they invest in also in the future. Being denied access to future versions of the software in which they have made investments would imply the loss of these investments.

Now consider how the two aspects—investment specificity and cumulativeness—interact. Cumulativeness entails that (a) the initial innovation(s) on which improvements build upon is (are) always totally or partly incorporated into the final output; or that (b) even if not incorporated, the initial innovation has to be available to developers/consumers in order for them to be willing to use/buy the program. As a consequence, if the original innovation input enjoys legal protection in the form of patents or copyright and unless its improver is given ex ante a defined set of rights to modify and redistribute modifications of the input, development through software-specific investment gives rise to a situation of holdup between the producer/owner of the input and the subsequent innovator.

Thus, the choice to open the source code under OSS licensing terms might be considered the choice of a contractual mechanism that provides *safeguards* against opportunism to all the potential contributors to development. Indeed, such contractual mechanism ensures that both the original and subsequent developers will have continued access to the software they use to produce services in the final market. In so doing, it allows the original developer to gather contributions from other developers similarly interested in increasing the value of a software they use to provide complementary services[8]. This turns out to be of particular relevance when the market value of a given software is declining and the original developer risks losing the specific investment made in the first place.

It is important to note that the assumptions we make on the choice to open the source code imply that we do not directly address the question whether software development under an OSS license is superior to software development under proprietary licenses *in absolute terms*. In other words,

[7] This is consistent with the fact that the business models currently adopted in the software industry are increasingly based upon the idea that it is service provision that should be sold rather than binary code (think about the increasing importance of the Software as a Service model—SAAS model).

[8] This is consistent with all of the empirical analyses available to date that confirm the relevance of user needs as the single most important driver of contributions to OSS projects. Lakhani and Wolf (2003) in a web-based survey administered to 684 software developers in 287 F/OSS projects find user needs, both work- and non-work-related, to be the overwhelming reason for contribution and participation. Similar results are obtained also by Gosh, Glott, Kreiger and Robles (2002); Hertel, Niedner and Hermann (2002)

we move from the empirically sound premise that the choice to open the source code may be compatible with the incentives of a self-interested individual under specific circumstances but we do not directly explore here the issue of whether the same or better results could be achieved through alternative means, such as for instance through a centralized organizational solution involving the hiring of developers to work in-house on the project.

Although a comprehensive treatment of this problem is outside the scope of this paper, it is possible to highlight some reasons why a decentralized solution such as recourse to OSS license-mediated collaboration may be chosen instead of a proprietary solution with centralized ownership. These reasons have to do, in particular, with the usual agency problems associated to employment contracts. A centralized solution involves a need to provide remuneration in order to motivate developers that, in turn, implies a need for effective selection of the most talented individuals. This determines not only selection and monitoring problems but also, and most importantly, a necessary reduction of the pool of developers. If it is important to ensure the participation of a large pool of developers with heterogeneous human capital these drawbacks may be decisive (on some aspects of this issue see, for instance, Johnson, 2006).

3. The choice of the OSS license: copyleft vs. non-copyleft licenses

As mentioned before, OSS licenses can be roughly said to belong to two types: copyleft and non-copyleft. The difference between the two types resides in the nature of the constraints they impose on licensees' freedom to redistribute the modified version of the OSS-licensed software under terms of his choice.

Non-copyleft licenses. This sort of licenses allows to release modifications of the software under a different license, even a proprietary one. Well known examples are the Berkeley Software Distribution (or BSD) license, the Apache License and the X11 license. The main obligation imposed by these licenses concerns the need to give credit to contributors. All that is needed for anyone to freely use non-copyleft-licensed software is to include in the redistributed software the copyright notices (one of the notices, in turn, requires that subsequent distributors also include the notice, so that it passes from user to user).

Software put in the public domain (i.e. software whose author has explicitly given up copyright) can be described, in terms of its implications for incentives, as a nonproprietary license with no associated constraints—

although in this case there is neither an owner nor a license.

Copyleft licenses. Copyleft licenses impose more stringent constraints relative to non-copyleft licenses. From our point of view, the most relevant of such additional constraints concerns the obligation to license future developments under the same terms. Thus, developers of contributions to the original software code retain copyright over their creations but they must distribute them under the terms of the initial license. This constraint is imposed, among others, by the General Public Licence, or GPL (see section 2(b) of the GPL)[9], which is therefore a copyleft license and on which we will focus in this paper.

Thus, copyleft and non-copyleft licenses differ in that the former significantly restrict both the original licensors' and licensees' freedoms with respect to the latter. The most relevant of these restrictions relates to the fact that the BSD (and in general all non-copyleft open source licenses) grants developers the freedom to develop the OSS-licensed software and license developments as proprietary, while the GPL does not. In other words, the BSD, differently from the GPL, grants developers the possibility to exclude other users and developers from access to an improved version of the software or from a software using the original one as a component. This implies that the BSD opens up the possibility to charge a price for access.

It is perhaps important to note that the wording of the GPL license does not prevent developers from charging a fee as high as they wish (or can) for the distribution of software. In fact, GPL licenses do not impose an express obligation to release the software itself for free, although they impose an obligation to apply the terms of the original license *at no charge to all third parties* (see, for instance, section 2(b) of the GPL)[10]. However, the latter provision, combined with the public good characteristics of software (and thus with the possibility to reproduce copies of software code at virtually zero cost by anyone), implies that the equilibrium price of the right of access to the software is likely to be zero.

Given that conventional wisdom has it that the broader the rights granted to an economic agent, the greater the stream of benefits she can derive from her property and therefore the greater her incentives, the absolute dominance

[9] Section 2(b) of the GPL reads:

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

[10] For more information refer, for instance, to the GPL FAQ webpage, available at http://www.gnu.org/licenses/gpl-faq.html.

of the GPL in quantitative terms appears puzzling. Given that the BSD grants a broader set of rights to both licensors and licensees we should observe a prevalence of the BSD over the GPL. Instead, copyleft licenses are much more widely adopted than non-copyleft licenses, in spite of the more stringent restrictions they impose. Lerner and Tirole (2005), for instance, report that 72% of OSS projects on the Sourgeforge database adopt a GPL license, while only 7% of the projects in their sample adopt the BSD.

Looking at "big" project, there are many well known examples of projects which turned from proprietary to a GPL licence: OpenOffice, Mozilla/Firefox, MySQL, Virtualbox, QEMU, OpenSolaris, Xsara Xtreme, the Qt libraries and (announced) Java. There are no similar examples of software turned to a BSD-like licence; the only case we know of software moved from the GPL to BSD is the OSS standard for music compression (an alternative to MP3).

Most of the explanations offered for why the GPL may work have to do with ideology, either in the sense that the GPL constitutes a means to ensure that the expectations of ideologically-motivated contributors are not frustrated by the commercialization of the result of their effort (see, for instance, Frank and Jungwirth, 2001) or in the sense that the GPL allows to attract ideologically-motivated contributions when other sources of motivation are weak (Lerner and Tirole, 2005). Other reasons have to do with GPL's ability to prevent forking (Maurer and Scotchmer, 2006) or to reduce the extent of free-riding, particularly in the form of the privatization of existing OSS projects (Gambardella and Hall, 2005)[11].

In fact, although we do not deny the importance of ideology, we contend that there is no reason why the dominance of the GPL should be explained only in terms of an ideological preference of developers. In our view, an important element of the explanation of the relative success in terms of diffusion of copyleft and non-copyleft licenses is the fact that the two types of licenses impact differently on developers' ability to access future versions of the software originally released under OSS terms. This, in turn, influences developers' investment choices. While both types of OSS licenses mitigate to some extent the mentioned hold up problem arising from the combination of cumulativeness and investment specificity relative to proprietary licenses, copyleft and non-copyleft licenses have very different implications in this regard. In particular, although copyleft licenses entail a narrower set of freedoms for both the original licensor and licensees, they guarantee to both

[11] The comparison of the costs and benefits of these two forms of licensing from an economic viewpoint has so far received scant attention (with the exception of Gaudeul (2005); Bezroukov (1999)). Indeed, while the literature highlights a number of reasons why recourse to the GPL may make sense, it does not explore the question whether the GPL may make more or less sense than the BSD and under what circumstances this is likely to be the case.

that they will have continued access to the software in whose knowledge they have invested in and in all future versions of it and therefore constitute a strong safeguard against the threat of hold up.

Non-copyleft licenses, by contrast, allow subsequent developers to turn their contributions into proprietary and thus expose those developers that stick to the OSS strategy to the the risk of opportunism and to a form of ex-post hold up of the specific investment in human capital they have made. This holds even if developers are not excluded from access to older versions of the software, as they might be excluded from the subsequent versions of the software, which may include contributions important from a technical and commercial viewpoint. This impacts both on the *level* and on the *nature* of the investment chosen ex-ante, as it will become clear in the following section.

4. A formal model of investment choice under different licenses

The previous section has explained that the principal difference between GPL and BSD licenses resides in the fact that the latter allows developers, both licensors and licensees, to license their developments under proprietary terms. Hence, if the choice between the BSD and the GPL is of some relevance, it is because with positive probability, at a certain future stage, the latest and most valuable version of a project under BSD can be excluded from access and no longer be open source. Since the possibility to exclude and make the project proprietary allows the developer to collect a price from other users and developers, here is where the superiority of the BSD in terms of incentives is alleged to reside.

In order to emphasize the difference between the BSD and the GPL we consider an "ideal" case where ex post contracting is without cost and there are no "social norms" in the community of developers that keep developers from excluding others and negotiating the conditions of access to their developments[12]. By focusing on a simple single-period model, we assume that after development has taken place all developers may choose to *sell* their contributions to others. We assume that bargaining is efficient, so that ex post each developer has access to each other's contribution. In so doing, we consider a case which is at the same time more favourable to the choice of the BSD than real world conditions are, and which makes the difference between the BSD and the GPL larger than it actually is.

We consider a software which is an input to the production by each de-

[12] The presence of social norms in the OSS community might be considered a substitute for licenses as a means of coordination. More on this will be said in section 5.

veloper in the final market. Agents do not profit from the direct sale of the software but rather from providing assistance, customization or other services to end-users and access to the source code constitutes a necessary input into the provision of such services, so that software development constitutes a byproduct of these activities, rather than the opposite.

4.1. Model setup

We assume that each developer i makes an investment y_i specific to software under an open source license, whose technical quality is represented by an index X. After the investment is made, she develops an innovation. Innovations increase the value of X for developers as they use X to produce services in their final market. Considering the set N of all developers, let X_S be the level of X when contributions by developers in S are included. Clearly, $X_S \geq X_R$ if $R \subset S$.

Let π^i be developer i's profit in the final market. We have $\pi^i = \pi^i(y_i, X)$, with

$$\frac{\partial \pi^i(y_i, X)}{\partial y_i} > 0 \qquad \frac{\partial \pi^i(y_i, X)}{\partial X} > 0 \qquad \frac{\partial^2 \pi^i(y_i, X)}{\partial X \partial y_i} > 0 \tag{1}$$

We allow for an effect of y_i on π independent of the effect through X in order to consider that (1) the investment by i in the development of X can increase her (X-specific) human capital, or can have a signalling effect on the final market; (2) the investment can improve the software in a developer-specific way, since developers can use "specialized" versions of X. In this sense, the effect of y_i on X must be thought of as the transferrable effect of y_i, the effect of y_i which benefits the whole community.

As mentioned in the previous paragraphs, the presence of the "direct" effect of y_i on π^i is very important in the explanation of why developers may choose an open source solution.

We consider the ex post interaction under the two cases of GPL and BSD license.

GPL. In the GPL case, contributions can be freely accessed by other developers; they will be included in X, and developers are rewarded for using X to provide services in the final market. The payoff of developer i is:

$$\pi^i(y_i, X_N) - y_i. \tag{2}$$

BSD. When instead developers are allowed to exclude other developers from access to their contribution, innovations are merged only after the innovator grants access to other developers. In order to make the innovation

available, the developer can ask a price, so that bargaining will take place among developers in order to allocate the surplus from innovations. Each developer's share in this surplus is determined by her bargaining power, which in turn is a function of how important is her own contribution.

Considering that bargaining is efficient, we will make use of the concept of Shapley value. The use of this concept has an established tradition in the economic analysis of incomplete contracts and property rights Hart and Moore (see, for instance, 1990). The Shapley value considers the share of a bargainer as a function of her value for each possible coalition of bargainers $S \subseteq N$.

The value for developer i is

$$\sum_{S \subseteq N \,|\, i \in S} \rho(S)\Big[\Pi(S) - \Pi(S \backslash \{i\})\Big]; \tag{3}$$

where

$$\rho(S) = \frac{(|S|-1)!(|N|-|S|)!}{|N|!} \tag{4}$$

and where $\Pi(S)$ is the total profit obtained by coalition S, or

$$\Pi(S) = \sum_{j \in S} \pi^j(y_j, X_S) \tag{5}$$

so that $\Pi(S) - \Pi(S \backslash \{i\})$ is how much the profit of coalition S is reduced if i leaves it.

The share represented by the Shapley value is a weighted average of the contribution of i's development to all possible subsets of developments[13]. The formula is often justified by imagining that the coalition N is formed one actor at a time, with each agent obtaining her contribution (as if she could make a take-it-or-leave offer to the agents already in the coalition), and then averaging over the possible different permutations in which the coalition can be formed[14].

4.2. The choice of the level of co-specificity

In addition to the choice of the investment *level* y_i, we consider as very important the choice of the *nature* of the investment in development. Each developer can choose to make her development based on X more or less co-specific to developments made by others. At one extreme, developer i's

[13] Note that $\sum_{S \subseteq N \,|\, i \in S} \rho(S) = 1$.

[14] Taking all possible orderings of $|N|$ agents as equally likely, $\rho(S)$ represent the probability that i will be ranked just after the agents in the set $S \backslash \{i\}$.

contribution can be stand alone and require only the basic version of the software. At the other extreme, her contribution can have value only if used together with all the other contributions. More generally, each developer can decide to make her development more or less specific to other developments.

Software development efforts are always to some extent co-specific. However, increasing the degree of co-specificity of efforts generally allows to reap important technical benefits, by enhancing the effectiveness of the combination of the various modules or functionalities. This is particularly important in the a context of decentralized innovation typical of OSS development, where typically each contribution is soon made available and "used" by other developers.

We will say that the contribution by i is specific to the subset of contributions $R \subseteq N$ if its value is enhanced by the fact that it is used together with the contributions in R, or

$$\frac{\partial X_S}{\partial y_i} > \frac{\partial X_{S'}}{\partial y_i} \qquad \text{when} \quad R \subseteq S \text{ and } R \not\subseteq S'. \tag{6}$$

We will say that a developer whose contribution is specific to R can increase the level of cospecificity by choosing to make it specific to R' such that $R \subset R'$; we assume that, as a consequence of increased cospecificity, $\partial X_S / \partial y_i$ is increased for all S such that $R' \subseteq S$, while it is decreased for S such that $(S \cap R') \subseteq R$. In other words, more cospecificity increases the value of the investment when all contributions in R' are included in S, while it decreases it when it is made specific to additional contributions which are not included in S.

A somehow extreme assumption is that i's R-specific contribution doesn't add anything to X_S ($\partial X_S / \partial y_i = 0$) if $R \not\subseteq S$.

Hence, there is a trade-off from increasing co-specificity (i.e. from choosing to make an investment specific to a larger set of contributions R') when we are not sure that all contributions in R' are included, while cospecificity is always good when all contributions in N are used.

When the objective is to maximize the development of X, co-specificity should always be encouraged, since more cospecificity means a higher X_N, and under the assumption of efficient ex post bargaining the coalition N will always be formed (i.e. all contributions will be included).

Since the degree of co-specificity is an individual choice of each developer, the license can affect this choice. Under the GPL, developers are sure coalition N is always formed, and take their investment decisions on the basis of this expectation. Under the BSD, conversely, the bargaining power of the parties depend on their "outside" options, i.e. the value with all coalitions different from N. In other words, under the BSD increasing the value

of coalitions different from S will be optimal from an individual point of view[15].

We summarize this first conclusion in the following

Proposition 1. *Under the GPL, each developers will choose to be specific to all other develpers (N-specific), i.e. the highest possible degree of co-specificity. Under a BSD license, a lower degree of co-specificity will be chosen.*

The second part of the proposition is best illustrated by using a simple two-agents specification of the model presented above. This allows to discuss the basic characteristics of the interaction in the simplest possible setting.

Let X be $X_{\{1,2\}} = \theta^1 y_1 + \theta^2 y_2$ when contributions are merged together, and $X_{\{i\}} = \bar{\theta}^i y_i$ when only i's contribution is used[16].

Depending on the choice of the parties with regard to the degree of co-specificity, θ^1 can assume the following values:

co-specific investment by:	both	only 1	only 2	none
contributions merged	θ^1_{12}	θ^1_1	θ^1_2	θ^1_0
contributions not merged	$\bar{\theta}^1_1$		$\bar{\theta}^1_0$	

Hence, we indicate by θ^i_{12} the marginal effect of y_i on X when both 1 and 2 choose to work on co-specific projects and the two contributions are merged; if the contributions are not merged, the effect is $\bar{\theta}^i_i$ (note that in this case it is not relevant if the other developer has chosen a co-specific investment or not). The remaining notation is intuitive.

The assumption above about the effects of incresed specificity translates into $\theta^i_{12} > \theta^i_j > \theta^i_0$: the choice of a co-specific investment increases the value of the investment when contributions are merged, and the value is maximum when both choose the co-specific investment. Although this is not necessary to our result, it might be reasonable to assume strategic complementarity in the choice to be co-specific: $\theta^i_{12} - \theta^i_j > \theta^i_i - \theta^i_0$.

If contributions are not merged, it is better not to make a co-specific investment, hence $\bar{\theta}^i_0 > \bar{\theta}^i_i$. We assume without loss of generality that $\bar{\theta}^i_i = 0$ $(i = 1, 2)$: the value of the contribution is zero if i has chosen to be co-specific and the contributions are not merged.

[15] Note that if we took into account transaction costs, hence the possibility that the coalition N is not formed in the end, this would only reinforce our point, as it would make it less important to increase X_N.

[16] Therefore, $\partial X_{\{1,2\}}/\partial y_i = \theta^i$ and $\partial X_{\{i\}}/\partial y_i = \bar{\theta}^i$.

Note that from a collective point of view the optimal choice is the co-specific investment, since in equilibrium all contributions are merged, and all contributors have access to $X_{\{1,2\}}$.

Let us compare the two licensing regimes of GPL and BSD.

Under the GPL, each developer has access to X after development activity has taken place. There is no reason not to contribute one's development. Each developer gets (2). The level of co-specificity will be chosen so that $X_{\{1,2\}}$ is maximized: the "socially" optimal outcome θ^i_{12} will result.

Things are different under the BSD. In this case, developers' payoffs depend on their contractual force, which in turn depends on the value of their contribution to *all* possible coalitions, not only to N (in the two-agents case, each developer can belong to a coalition with the other developer or simply stay alone). According to the solution defined in (3), which in the two-agent case coincides with the Nash bargaining rule (the parties split 50:50 the difference between the payoff with cooperation and the payoff when they do not cooperate), i gets

$$\frac{1}{2}\left[\pi^1(y_1, X_{\{1,2\}}) + \pi^2(y_2, X_{\{1,2\}}) - \pi^1(y_1, X_{\{1\}}) - \pi^2(y_2, X_{\{2\}})\right] + \pi^i(y_i, X_{\{i\}}) \tag{7}$$

or, for developer 1:

$$\frac{1}{2}\left[\pi^1(y_1, X_{\{1,2\}}) + \pi^2(y_2, X_{\{1,2\}}) - \pi^2(y_2, X_{\{2\}})\right] + \frac{1}{2}\pi^1(y_1, X_{\{1\}}) \tag{8}$$

with a similar payoff function for developer 2.

We now consider the optimal choice of the level of co-specificity by the parties. To simplify the analysis and make our conclusion more clear-cut, we disregard the choice of y_i assuming $y_i = 1$. We will remove this restriction in the following section.

Substituting for X in the payoff function (7), and defining for notational convenience $\hat{\pi}(X) = \pi^1(X) + \pi^2(X)$, we can consider the (noncooperative Nash) equilibrium of the choice of co-specificity by the parties.

Consider the case that developer 2 chooses co-specificity. Developer 1 will choose co-specificity only if

$$\frac{1}{2}\pi^1(\bar{\theta}^1_0) + \frac{1}{2}\left(\hat{\pi}(\theta^1_2 + \theta^2_2) - \pi^2(0)\right) \tag{9}$$

is lower than

$$\frac{1}{2}\pi^1(0) + \frac{1}{2}\left(\hat{\pi}(\theta^1_{12} + \theta^2_{12}) - \pi^2(0)\right) \tag{10}$$

or

$$\pi^1(\bar{\theta}^1_0) - \pi^1(0) < \hat{\pi}(\theta^1_{12} + \theta^2_{12}) - \hat{\pi}(\theta^1_2 + \theta^2_2) \tag{11}$$

On the other hand, if developer 1 does not choose co-specificity, developer 2 will choose co-specificity only if

$$\frac{1}{2}\pi^2(\bar{\theta}_0^2) + \frac{1}{2}\left(\hat{\pi}(\theta_0^1 + \theta_0^2) - \pi^1(\bar{\theta}_0^1)\right) \tag{12}$$

is lower than

$$\frac{1}{2}\pi^2(0) + \frac{1}{2}\left(\hat{\pi}(\theta_2^1 + \theta_2^2) - \pi^1(\bar{\theta}_0^1)\right) \tag{13}$$

or

$$\pi^2(\bar{\theta}_0^2) - \pi^2(0) < \hat{\pi}(\theta_2^1 + \theta_2^2) - \hat{\pi}(\theta_0^1 + \theta_0^2) \tag{14}$$

Therefore co-specificity, though efficient, might not be an equilibrium strategy under the BSD. This will be the case if $\pi^i(\bar{\theta}_0^i) - \pi^i(0)$ is high enough with respect to the gains from co-specificity.

A numerical example To show that the efficiency loss can be substantial, consider the following example with $n+1$ developers, with a large agent (named $n+1$) and n small agents. By "large" and "small" we mean, respectively, "with a large potential contribution" and "with a small potential contribution". We assume that coalitions of small agents not involving the large one give no advantage in terms of co-specificity, so that this example can be seen as a straightforward extension of the simple two-agents case.

Assume that, when it is merged into X, developer i's ($i < n$) contribution to $\hat{\pi}$ is equal to:
- 24 both agent i and agent $n+1$ have chosen co-specificity;
- 20 if i has chosen co-specificity but $n+1$ has not;
- 12 if neither i nor $n+1$ has chosen co-specificity.

We are assuming that developers other than $n+1$ do not affect developer i's contribution.

Assume that π^i (stand alone profit for a small developer) is equal to 10 when he chooses not to be co-specific, zero when co-specificity is chosen.

Finally, assume that by not being co-specific the stand alone profit π^{n+1} is higher by F than if she chooses to be co-specific, with $F > n \times 2$.

Consider the case that all n small developers choose co-specificity. The total profit is $n \times 24$ if $n+1$ is co-specific. However, this is not optimal for $n+1$, since her payoff is $\frac{1}{2}\hat{\pi} + \frac{1}{2}\pi^{n+1}$ and she can increase π^{n+1} by F by decreasing $\hat{\pi}$ by $n \times 2 < F$ (this is the effect of choosing not to be co-specific).

However, if $n+1$ chooses not to be co-specific, each i must compare his profit when co-specific ($\frac{1}{2}20$) and when not co-specific ($\frac{1}{2}12 + \frac{1}{2}10$). He will choose not to be co-specific, and this will be the only Nash equilibrium of the game.

Note that, in the example, there is no real advantage from merging the contributions: the total profit is $n \times 12$ if contributions are merged, and $n \times 10 + F$ if they are not. In any case, it is much lower than if co-specificity is chosen by all developers, in which case we would have $n \times 24$.

4.3. The choice of the level of investment

The analysis of the previous sections disregarded the incentive to invest, i.e. the choice of y_i. Although the conclusion of proposition 1 is not affected by this simplification, the comparison between the two licenses is about the level of X and of π^i, and the level of y_i is relevant in this regard. Moreover, recall that it is because it allegedly induces a higher level of y_i that the BSD is often considered superior to the GPL in terms of incentives, so that it is important to consider this aspect explicitly.

In the GPL case, the value of a marginal increase in y_i is:

$$\frac{\partial \pi^i(y_i, X_N)}{\partial y_i} + \frac{\partial \pi^i(y_i, X_N)}{\partial X}\frac{\partial X_N}{\partial y_i} - 1 \tag{15}$$

developers will invest as long as this is higher than zero.

In the case of the BSD, the marginal effect of an increase in y_i is (we differentiate (3)):

$$\sum_{S \subseteq N | i \in S} \rho(S) \left[\frac{\partial \pi^i(y_i, X_S)}{\partial y_i} + \sum_{j \in S} \frac{\partial \pi^j(y_j, X_S)}{\partial X} \frac{\partial X_S}{\partial y_i} \right] - 1 \tag{16}$$

Comparing this expression with (15), we notice that:

- the incentive to invest due to the direct (hydiosincratic) effect of y_i on i's profit on the final market, is lower under the BSD as, because of (1),
$$\frac{\partial \pi^i(y_i, X_N)}{\partial y_i} > \sum_{S \subseteq N | i \in S} \rho(S) \frac{\partial \pi^i(y_i, X_S)}{\partial y_i}; \tag{17}$$
- in the BSD case, each developer reaps a share of the benefits of her development on the profits of all developers. If this is higher than the benefit on π^i only, or
$$\frac{\partial \pi^i(y_i, X_N)}{\partial X}\frac{\partial X_N}{\partial y_i} < \sum_{S \subseteq N | i \in S} \rho(S) \sum_{j \in S} \frac{\partial \pi^j(y_j, X_S)}{\partial X} \frac{\partial X_S}{\partial y_i} \tag{18}$$
then the BSD scores better in this regard.

Under the GPL, the incentive to invest is given by the perspective to use the software in one's final market. Development is somehow a byproduct of the investment to enhance one's X-specific skills and to introduce those improvements in X that affect most π^i.

Under the BSD, the incentive to increase π^i is less important, but there is the opportunity to "sell" innovations to other developers.

Although it is not possible *in general* to conclude that the incentives to invest by those who participate to the development of X is higher under one system or the other, the presumption is that in many cases the second of the two effects is more important, and the BSD might induce a higher level of y_i.

It is worth emphasizing that even when the BSD induces a higher y_i, it is not possible to draw a conclusion on the superiority in general of one license of the other, since the overall effect of investments on X (hence on Π), depends on the combined effect of the choice of the nature of the investment (more or less co-specific) and the intensity of investments.

However, a conclusion can be reached in some special cases.

Note that the inequality (18) is less likely to be verified the lower are the terms $\partial X_S/\partial y_i$ with respect to $\partial X_N/\partial y_i$, i.e. the more important is the effect of co-specificity.

This suggests what are the circumstances when the GPL induces a higher y_i. Once again, co-specificity can play a central role.

Consider the case in which developers choose to make a co-specific investment, and make the assumption that their investment is valuable only if the coalition N is formed (note that, under these circumstances, the choice of a co-specific investment is an equilibrium both with GPL and with BSD). In other words, we are considering that $\partial X_S/\partial y_i = 0$ for $S \subset N$. In the two agents case, this amounts to the assumption that $\bar{\theta}_i^i = 0$.

From (16), we have that the first order condition with regard to the choice of y_i is now:

$$\frac{1}{|N|}\,\frac{\partial \pi^i(y_i, X_N)}{\partial y_i} + \frac{1}{|N|}\sum_{j\in N}\frac{\partial \pi^j(y_j, X_N)}{\partial X}\,\frac{\partial X_N}{\partial y_i} = 1 \tag{19}$$

The second term on the left hand side represents the average effect of an increase in y_i on the individual profit of developers. Depending on the cases, this can be higher or lower than the second term in the expression (15).

By comparing (19) with (15), we realize that the level of y_i is likely to be higher under the GPL for most i. The loss in incentives (with respect to the GPL) is higher the more important is the effect of y_i on π^i, $\partial \pi^i(y_i, X_N)/\partial y_i$.

A similar result is obtained under less extreme hypotheses on $\partial X_S/\partial y_i$,

if we assume that the use of the software is profitable only when the technological index X_S reaches a certain threshold level. Hence, if we assume that $\pi^i(X) = 0$ for $X < \bar{X}$ and $X_N \geq \bar{X}$ only when the co-specific investment is chosen, the first order condition is once again (19), and the GPL will be more attractive in terms of incentives to invest.

Such an assumption may be justifiable in contexts where the degree of technological advancement of X is the crucial variable to secure profitability. We expect this can be the case, for instance, in a market where competing projects struggle for technological leadership. This is consistent with the observation that the fact that a software is "lagging behind" is often a reason why the open source solution is chosen, in the hope of spurring technological advancement.

We summarize the conclusion of this section in the following

Proposition 2. *Taking into account incentives to invest, the BSD license can be expected to induce a higher level of y_i. However, the more important is the effect of co-specificity on X or on Π, the lower is the chance that y_i will be higher under the BSD.*

5. Discussion

The model introduced in the previous section may help to shed light on three stylized facts. The first is the mentioned quantitative dominance of the GPL license in the OSS world. The model suggests that, although the BSD might have an advantage over the BSD in terms of incentives to expend effort in software development, it tends to distort the nature of development, by providing sub-optimal incentives to make investments co-specific to those of other developers. This, in turn, suggests a reason why the GPL license might be preferred to the BSD by a developer interested in maximizing the subsequent rate of development, as we claim it is the case for many project started as or moved to the open source.

The second is the fact that there tends to be a correlation between the type of OSS project and the type of license adopted. BSD-like licenses seem to be more widespread for setting standards that give rise to the articulation of different projects, and they are often used for projects funded by third parties (government, universities etc.)[17]. They are practically never observed for end-user projects—i.e. for projects meant to produce a software that can be directly used by final customers. GPL licenses are, by contrast, routinely adopted for the latter type of projects. An interesting example that may

[17] Another case is the development of drivers for peripherals.

confirm this observation is given by the GNOME project–aimed at building a complete desktop environment–that contains software packages licensed under both BSD and GPL. Within this project, not a single end-user application is licensed under BSD, although there is no restriction on the choice of the kind of license to use. Consistently with these empirical observations, our model identifies the differential advantages of the two licenses, pointing out that:

1. the GPL is better suited than the BSD to coordinate and encourage joint effort by many (possibly small) developers; while

2. the BSD is better suited than the GPL to generate positive spillovers to other developers, when no feedback is required.

Indeed, standard-setting requires a relatively limited amount of feedback and it is generally performed by a limited number of agents. By contrast, end-user applications tend to require a much greater amount of co-specific investment.

Finally, a third relevant but relatively unnoticed empirical regularity is the fact that BSD and GPL communities tend to be rather different. In particular, BSD communities tend to be close-knit groups of developers who work on projects where feedbacks from outside the relatively stable community are limited and tend to rely on strong social norms, repeated interaction as well as relatively structured coordination mechanisms other than the license itself (one example is the adoption of a voting committee of co-developers within the Apache community, which uses a variant of the BSD license; another example is the setting up of a consortium within the X11 project). GPL communities, by contrast, tend to involve a higher number of participants and a less prominent role of social norms.

In this regard, our model might be taken to suggest that, if co-specificity of development efforts is important for a given BSD-based project, the license alone may not be an effective coordination mechanism, so that other means of coordination should come into play. Thus, within BSD communities the major role played by "face-to-face" interaction or the adoption of other formal coordination arrangements may make the license of secondary importance in securing co-specific effort, while at the same time a less constrained license can encourage independent investments by other developers. The GPL is, by contrast, a better choice when development feedbacks are important in a context where relations between developers are more "anonymous".

As a final note, it is perhaps important to stress that, in spite of the important differences in the wording of the license, projects under BSD and under GPL may show a similar degree of "persistence" as open source projects. In

other words, differently from the assumptions we make in the model, real world developers, though allowed to do so, can decide not to exclude other developers from access to their contributions. This can be explained by the existence of social norms and of other coordination mechanisms and ideological reasons can play a role, as well as by the fact that the conditions that made the choice of an OSS strategy the best solution at the first stage may be still valid at subsequent stages for all developers. A complementary reason is that, contrary to what we assumed, there might be high contracting and marketing costs, and this is expecially true when the development is of small importance and/or the contributor must incur high fixed costs to market and enforce his property rights. When ex post contracting costs are very high, the BSD behaves like a GPL[18].

Future research might take into explicit account contracting costs. However, we think that the basic result in terms of differences between the two licences should not be affected.

References

Aghion, P., Tirole, J., 1994. "On the management of innovation". *Quarterly Journal of Economics*, vol. 109, pp. 1185–1207.

Arora, A., Merges, R. P., 2001. "Property rights, firm boundaries and r&d inputs". mimeo, Carnegie Mellon University and U.C. Berkeley School of Law.

Bergquist, M., Ljungberg, J., 2001. "The power of gifts: organizing social relationships in open source communities". *Information Systems Journal*, vol. 11, pp. 305–320.

Bezroukov, N., 1999. "Open source software development as a special type of academic research: critique of vulgar raymondism". *First Monday*, vol. 4.

Cai, H., 2003. "A theory of joint asset owneship". *RAND Journal of Economics*, vol. 34, pp. 63–77.

Frank, E., Jungwirth, C., 2001. "Reconciling investors and donators—the governance structure of open source". Working paper, University of Zurich.

Gambardella, A., Hall, B. H., 2005. "Proprietary vs. public domain licensing of software and research products". Working Paper 11120, NBER.

[18] When they are high but not high enough, contracting takes place, though some cost is paid. The project becomes proprietary and transaction costs reduce the payoff.

Gaudeul, A., 2005. "Public provision of a private good: What is the point of the BSD license?" URL http://ideas.repec.org/p/wpa/wuwpio/0511002.html.

Gosh, R. A., Glott, R., Kreiger, B., Robles, G., 2002. "The free/libre and open source software developers survey and study". URL http://www.infonomics.nl/FLOSS/report.

Grossman, S. J., Hart, O. D., 1986. "The costs and benefits of ownership: a theory of vertical and lateral integration". *Journal of Political Economy*, vol. 94, no. 4, pp. 691–719.

Hart, O. D., 1995. "Corporate governance: some theory and implications". *Economic Journal*, vol. 105, pp. 678–89.

Hart, O. D., Moore, J., 1990. "Property rights and the nature of the firm". *Journal of Political Economy*, vol. 98, pp. 1119–1158.

Hertel, G., Niedner, S., Hermann, S., 2002. "Motivation of software developers in the open source projects: an internet-based survey of contributors to the Linux kernel". *Research Policy*, vol. 327, pp. 1159–1177.

Johnson, J. P., 2002. "Open source software: public provision of a public good". *Journal of Economics and Management Strategy*, vol. 11, no. 4, pp. 637–62.

Johnson, J. P., 2006. "Collaboration, peer review and open source software". *Information Economics and Policy*, , no. 18, pp. 477–497.

Lakhani, K., Wolf, R. G., 2003. "Why hackers do what they do: understanding motivation efforts in free/open source projects". Working Paper 4425-03, MIT Sloan School of Management.

Lerner, J., Tirole, J., 2002. "Some simple economics of open source". *Journal of Industrial Economics*, vol. 52, pp. 197–234.

Lerner, J., Tirole, J., 2005. "The scope of open source licensing". *Journal of Law Economics and Organization*, vol. 21, no. 1, pp. 20–56.

Maurer, S. M., Scotchmer, S., 2006. "Open source software: the new intellectual property paradigm". In: Hendershott, T. (ed.), *Handbook of Economics and Information Systems*, Elsevier, Amsterdam.

Moglen, E., 1999. "Anarchism triumphant: free software and the death of copyright". *First Monday*, vol. 48.

Raymond, E. S., 1998. "The cathedral and the bazaar". *First Monday*, vol. 330.

Rossi, M. A., 2006. "Decoding the Open Source puzzle: a survey of theoretical and empirical contributions". In: Bitzer, J., Schroder, P. (eds.), *The economics of Open Source Software development*, Elsevier, Amsterdam.

von Hippel, E., 2002. "Horizontal innovation networks: by and for users". Tech. Rep., MIT Sloan School of Management.